中国研究生创新实践系列大赛

“华为杯”第十七届中国研究生

数学建模竞赛

学　　校	东南大学
参赛队号	20102860127
队员姓名	1.李典泽
	2.付银
	3.程鑫

中国研究生创新实践系列大赛

“华为杯”第十七届中国研究生

数学建模竞赛

题 目 面向康复工程的脑电信号分析和判别模型

摘 要：

脑电信号的识别和分类是脑机接口技术中非常重要的一环，使用者无需通过复杂训练就可以获得较高的识别准确率，具有稳定的锁时性和高时间精度特性。本文使用**基于监督学习的随机森林，SVM 算法，半监督学习 S3VM** 等算法，以较高的分类准确度，完成了对 P300 脑电信号的分析和判别。

针对问题一，我们设计了 0.1~20Hz，阻带增益-80dB 的 46 阶带通巴特沃斯 IIR 滤波器对冗余数据做预处理，以是否出现 P300 脑电信号作为输入标签。考虑到发生 P300 事件和不发生的样本数不平衡，采用**随机排列抽取法**生成训练集，并建立了**基于随机森林算法的监督学习分类模型**。在 python 环境下搭建算法，预测 5 轮测试数据，采取“投票”机制得到不同受试者的 10 个测试结果（共 48 个目标字符），目标字符识别准确率为 **92%**，相比 KNN 模型 84%的准确率，识别效果更好。

针对问题二，在问题一数据处理的基础上，从闪烁时刻起记录有效训练数据，并生成一个维度为7500×20的矩阵，同样地对 P300 电位做标签化处理，得到一个 7500 维的向量。将通道选择问题转换成回归模型，进一步引入“spike and slab”先验分布，建立**贝叶斯稀疏回归模型**。使用 python3.8 实现 EP 算法，推导出模型后验分布近似分布，依据支撑向量后验分布参数选择出最优通道组合，基于通道选择结果，再次采用问题一的算法和模型，找到了对所有被试者都适用的一组最优通道组合：**Fz, F3, F4, C3, Cz, C4, CP3, CP4, CP5, CP6, P3, P4, P7**。

针对问题三，在问题一数据处理和问题二通道选择的基础上，选择 10 个字符的数据作为训练集（共$12\times5\times10\times5=3000$个样本），其中前 30%作为有标签数据集，后 70%作为无标签数据集，剩下作为验证集（共$12\times5\times2\times5=600$个样本）。建立**半监督学习的 S3VM 分类模型**，并引入正负样本比例因子 r 排除正负样本数目不平衡的问题。模型通过带标记样本预训练分类器，用拉格朗日启发式方法对无标签样本进行标签预测，扩充训练集。通过模拟退火的最优化方法进行迭代，并在训练时对初始无标签样本和有标签样本赋予不同的权重。最终在验证集上的平均准确率达到 **75.3%**，在测试数据中得到的结果基本吻合。

针对问题四，我们根据附件 2 所给的精确无噪声的样本标签数据，以及四个分区 Alpha，Beta，Theta，Delta 所占能量强度不同的特点，分别建立随机森林，支持向量机（SVM）,以及 LightGBM 三种监督学习算法训练睡眠预测模型。在测试集和训练集的选取

上，采用“**交叉验证法**”，克服因为训练集测试集不同分割方法引起模型性能不稳定。测试选取不同比例训练集数据对模型预测精度影响。综合比较 SVM，随机森林和 LGBM 算法预测精度和时间复杂度，得出，**LGBM 算法**得到的精度为 **89%**，且算法测试运行时间较短，其综合运行性能最优。

综上所述，本文建立的脑电信号分析和判别模型地实现了 P300 脑-机接口对目标字符的识别，最优通道选择，解决了带标签样本较难获得的问题。基于睡眠脑电信号频谱能量不同特点，建立模型实现了对睡眠状态的预测，有望成为评估睡眠质量、诊断和治疗睡眠相关疾病的重要辅助工具。

关键词：P300 脑机接口；监督学习；随机森林算法；通道选择；贝叶斯稀疏回归模型；EP；spike and slab；半监督学习 S3VM 算法

目 录

1．问题重述

1.1 问题背景

脑-机接口（Brain-Computer Interface,BCI）是一种在人脑与外部设备之间建立连接的系统。作为一项神经工程领域中最活跃且潜力无穷的新兴技术，在生物医学、神经康复和智能机器人等领域具有重要的研究意义的巨大的应用研发潜力。

P300 是一种由小概率事件刺激后约 300ms产生的诱发事件相关电位，是研究最多、应用最广泛的脑电信号，具有特定的目标性和可解释性，也是测试认知功能最常用的指标之一。P300 电位作为一种内源性成分，它不受刺激物理特性影响，与知觉或认知心理活动有关，与注意、记忆、智能等加工过程密切相关。基于P300 的脑-机接口优点是使用者无需通过复杂训练就可以获得较高的识别准确率，具有稳定的锁时性和高时间精度特性，因此，P300-BCI是最适合严重残疾患者独立长期在家庭环境下做康复训练使用的BCI系统。由于P300 脑机接口存在字符分类识别正确率不高以及信息传输率较低的问题，所以创建合适的模型算法来提高系统的精度、速度和能力是研究的核心目标。

1.2 问题提出

问题一：数据处理与目标分类问题。在考虑目标的分类准确率并保证信息传输速率的情况下，根据附件 1 所给数据，使用小于等于 5 轮次测试数据，找出附件 1 中 5 个被试测试集中的 9 或 10 个待识别目标，并给出具体的分类识别过程，可与几种方法进行对比说明设计方法的合理性。

问题二：数据选择与通道优化问题。设计一个通道选择算法处理原始冗余信息和通道数据，在满足通道组合数量的基础上给出针对每个单独被试、更有利于分类的通道名称组合。基于通道选择的结果，进一步分析对于所有被试都较适用的一组最优通道名称组合，并给出具体分析过程。

问题三：半监督学习分类问题。为了减少训练时间，请根据附件 1 所给数据，选择适量的样本作为有标签样本，其余训练样本作为无标签样本，在问题二所得一组最优通道组合的基础上，设计一种学习的方法，并利用问题二的测试数据（char13-char17）检验方法的有效性，同时利用所设计的学习方法找出测试集中的其余待识别目标（char18-char22）。

问题四：睡眠分期预测问题。根据附件 2 中所给的特征样本，用尽可能少的训练样本设计一个睡眠分期预测模型，并得到相对较高的预测准确率，给出训练数据和测试数据的选取方式和分配比例，说明具体的分类识别过程，并结合分类性能指标对预测的效果进行分析。

2．问题假设

（1）假设五名被测试成年人身体健康，无精神疾病史、脑部疾病史且视力正常；
（2）假设设备确定能够采集到P300 电位
（3）假设两次目标刺激产生的正电位波峰不会重合；
（4）假设个体间的差异不会超过设定值；

3．符号说明

$P_{i,j}$	判断是否出现 P300 电位的输入值
E	波峰出现的区间集合
F_S	采样频率
$t_{i,j}$	第i个目标字符在第j轮的起始采样时间
d_i	20 个通道一次采样数据的样本向量
t_k	闪烁起始时刻
N	一组样本中每个通道的采样点数
D	维度为7500×20的有效数据矩阵
z	支撑向量
μ	支撑向量的后验分布不等于 0 的概率
$\{x_i,y_i\}_{i=1}^{l}$	带标记的脑电数据样本集
$\{x_i\}_{i=l+1}^{n}$	未标记的脑电数据样本集
l	有标记样本集大小
u	无标记样本及大小
k	第二问中选取的k个最优通道
N	一组样本中每个通道的采样点数
r	正负样本个数比
x_i	第i行睡眠特征数据或通道权重
$\varphi(x_i)$	映射后的特征向量

4．问题一的模型建立与求解

4.1 问题一的分析

从问题的描述可知，在小概率刺激发生后 300 毫秒范围左右 P300 将会出现一个正向的波峰（根据 P300 的时域特征）。考虑到个体间的差异性，我们首先假设 P300 的波峰特征出现在目标符号所在行列闪现后的 300 毫秒左右的可偏差范围内。本题在预处理不同通道采集信号的基础上，结合是否在规定区间内出现 P300 脑电信号波峰特征为特征向量（分类标准），考虑到训练数据种类的不平衡，需要平衡 P300 和非 P300 电位的数据样本，并以此形成训练集并构建决策树，训练随机森林算法。使用算法预测 P300 信号出现的行和列，通过行列位置最终确定出 S1,S4,S5 的 10 个和 S2,S3 的 9 个待识别字符。

4.2 模型建立

4.2.1 数据降噪

本题附件给出数据是由传感器采集的原始的信号，由于电子元器件自身噪声，环境的影响以及被试者动作状态的影响，比如眨眼，皱眉头等，都会给采集数据带来一定的干扰，这些干扰大多为高频信号干扰。不正确的处理干扰数据会给接下来的模型训练带来极大的

影响。一般情况下，人体 P300 脑电信号的能量集中在 0.1Hz 到 20Hz 频带内，将原始数据经过通带为 0.1Hz 到 20Hz 的带通滤波器既可以滤除掉大多数高频噪声并且保留有效信息。我们基于 Matlab2018b 设计通带为 0.1Hz 到 20Hz，阻带增益-80dB 的 46 阶带通巴特沃斯 IIR 滤波器，其幅频特性如图 4-1 所示：

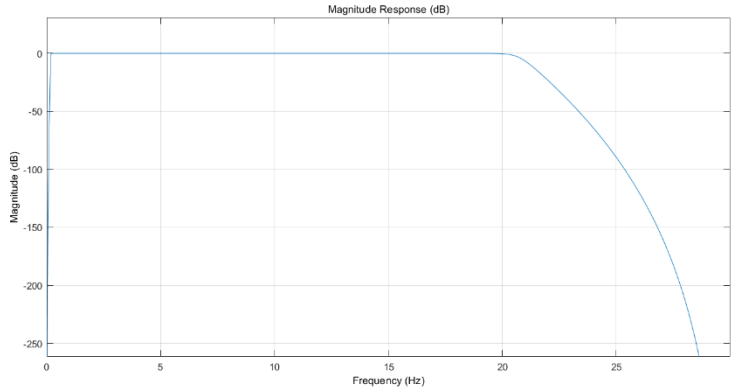


图 4-1 巴特沃斯 IIR 滤波器幅频特性图

选取 1 号被试者要求注视字符‘B’时，1 号通道第 250 个采样点往后 250 个采样序列为例，图 4-2 为未经滤波的信号，图 4-3 为经过带通滤波器滤除干扰后的信号。可以明显的看出原始信号“杂乱无章”的干扰被去除，经过滤波后的信号在第 46 个采样点附近出现疑似 P300 信号，查阅 1 号测试者的 train_event 数据发现，第 46 个采样点附近正好是字符‘B’的所在行闪烁的时候，此刻被试者 1 大脑中产生 P300 信号。说明设计的滤波器很好的滤除掉了干扰，保留了有效信息。

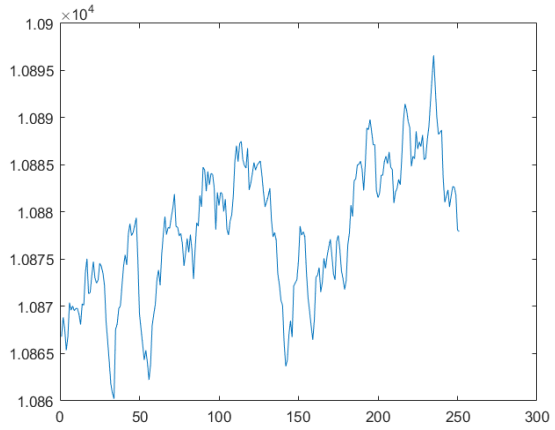


图 4-2 带通滤波器未经滤波的信号图

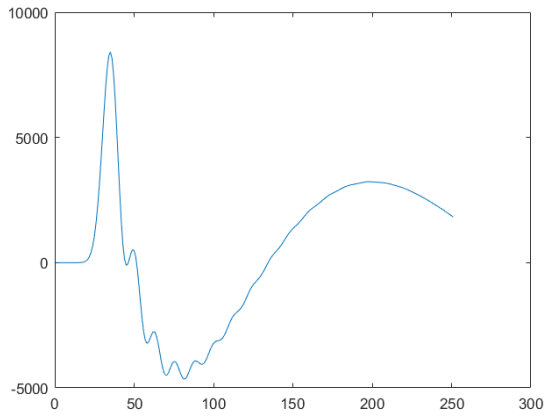


图 4-3 带通滤波器滤除干扰后的信号图

4.2.2 冗余数据去除

滤波虽然可以消除随机干扰带来的影响，但题目所给的数据是由 20 通道，250Hz 的采样频率得到，数据量仍然十分庞大，原始数据中必然包含大量的冗余信息，冗余信息不仅会加大系统传输处理数据的负担，甚至会对后期字符预测模型的训练产生严重影响，降低模型的预测精度。题干中描述在小概率刺激发生后 300 毫秒范围左右 P300 将会出现的一个正向波峰，考虑到个体间差异性，且为了保证能够完整包含该 P300 波峰特征成分， 假设 P300 的波峰特征出现在目标符号所在行列闪现时刻t_k后的$(t_k, 500 + t_k)$区间内，又采样点数和时间间隔成正比，即

$$N = F_S \cdot \Delta t \tag{1}$$

即有效信号仅会发生在目标所在行和列闪烁后的 125 个采样点内，而目标信号闪烁之前人脑并不会产生 P300 信号，因此采样闪烁之前的信号是没有意义的，不仅增加了数据冗余，还会干扰随后预测模型的精度。当然，我们不仅要采集目标字符出现后 125 个采样点这些有效数据，也需要非目标字符所在行或列闪烁后的 125 个采样点后的信号，这个数据作为人体不出现 P300 信号时候的正常脑电波信号，以此作为负的训练样本。每个行和列的闪烁时间为 80ms，闪烁后间隔 80ms 进行下一次闪烁，因此数据选取窗口的滑动点数大小应为 250Hz*(80+80)ms=40，数据的选取如下图所示：

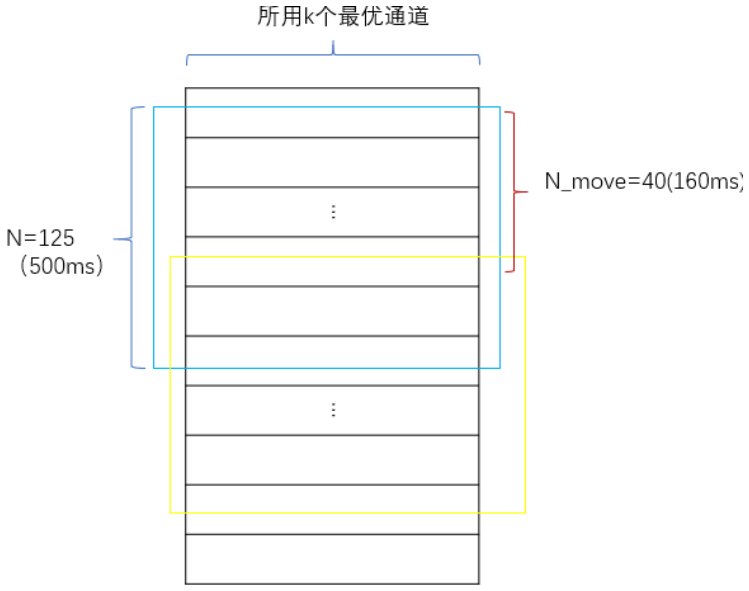


图 4-4 数据采样窗长示意图

4.2.3 数据标签的制作

预处理的另外一个重要的问题就是标签的制作。使用 4.2.1 以及 4.2.2 数据预处理及冗余去除过程后，我们仅仅需要判断当一个行或列闪烁时所采集的信号，判断 P300 信号是否出现，这里 P300 信号只有两种状态，有和无，所以将脑电信号的输入标签化处理：

$$P_{i,j}=\begin{cases}1 & \text{区间}E\text{中出现波峰特征}\\ 0 & \text{区间}E\text{中没有波峰特征}\end{cases}$$

其中，$P_{i,j}$用于判断是否出现 P300 电位，$E\in(t_{i,j},t_{i,j}+500)$，$t_{i,j}$表示第 i 个目标字符在第 j 轮的起始采样时间。

4.2.4 基于随机森林的分类问题算法

随机森林（Random Forest, RF）算法是一种组合分类器算法，包含许多独立的决策树（Decision Tree, DT）[6]。决策树算法以树状结构表示数据分类的结果，是一种典型的数据分类器。决策树是随机森林的核心部分，依据条件熵和信息增益保证决策树的左右生成，其基本原理为[7]：

（1）采用$Bagging$的方法从样本数N的数据集D中随机地选取不同变量数，用以形成众多的随机决策树并决定每棵树的分类节点。

（2）计算数据集D的信息熵，以及各个变量X_i和D之间的条件熵。信息熵$H(D)$体现了数据发生的不确定性，信息熵的值越高，表明越混乱，分类效果不好。条件熵$H(D|X_i)$是在数据集D发生的前提下，X_i变量所带来的熵值变化。用公式可表示为：

$$H(D)=E[-\log D_i]=-\sum_{i=1}^{n} D_i \log_2 D_i \tag{2}$$

$$H(D|X_i)=H(D,X_i)-H(X_i) \tag{3}$$

其中，$H(D,X_i)$为数据集D同变量之间的联合熵:

$$H(D,X_i)=-\sum_{i=1}^{n}\sum_{j=1}^{n} p(D,X_i)\log(D,X_i) \tag{4}$$

（3）计算信息增益$g(D,X_i)$，以表示数据不确定性的减小程度，计算公式为：

$$g(D,X_i)=H(D)-H(D|X_i) \tag{5}$$

（5）以信息熵为度量生成熵值下降最快的树，同时选择信息增益率和*Gini* 系数最大的变量作为决策树分裂节点的变量。*Gini* 系数的计算公式为：

$$Gini(p)=1-\sum_{k=1}^{n}\left(\frac{|C_k|}{|D|}\right)^2 \tag{6}$$

（6）每一棵决策树会根据待分类的数据给出一个基于该分类树的分类结果，并最终汇总全部森林里重复度最高的决策树结果（取众数）作为随机森林算法的分类结果。

用随机森林算法解决分类问题的流程如下：

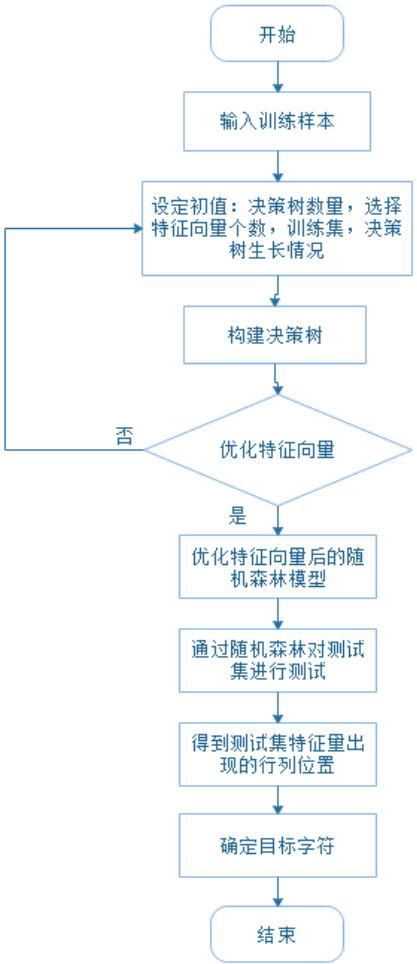


图 4-5 *RF* 算法流程图

4.3 模型的求解

4.3.1 平衡训练数据种类

使用 4.1 预处理数据训练模型时，存在一个非常严重的问题。每个被试者进行的每个字符的每轮实验中，每个行和列都会闪烁一次，即一共闪烁 12 次，但在每轮实验中，出现目标字符的行和列一共只有 2 个。因此，经过预处理后标签为 1 的数据，即出现 P300 波形的数据仅有 16.7%，其余 84%的数据的标签均为 0，即没有 P300 信号。用这样一个标签极其不平衡的数据训练的模型，最后模型必然出现预测结果大部分为 0 的情况。模型只要预测结果都为 0，训练精度就能达到 84%。

为了解决不同标签数据量不平衡问题，我们采用随机排列抽取法来降低标签为 0 数据量，保留标签为 1 数据量的方法。具体做法如下：每个被试者在进行同一个字符的 5 轮实验中，每轮实验目标字符所在行和列闪烁时所采集的数据，即标签为 1 的数据保留。剩下的行或列闪烁时，在每轮实验均分别取两个不重复的行或列作为有效训练数据。比如：第一个被试者在注视字符‘B’时，5 轮实验中，每次行 1 或者列 8 闪烁时的数据保留，然后取其采样点后 125 个数据作为有效数据，经过滤波，标签为 1。如预处理部分所述，共得到 10 组标签为 1 的有效信号。剩下共 10 个行或者列闪烁时分别在不同组取，如 2，3 行闪烁时采集的标签为 0 的信号取与实验 1，行 4，5 闪烁时候，经过预处理后标签为 0 的数据取于实验 2，以此类推。其余字符闪烁时也做同样的处理。这样每个实验者，在进行每个不同字符的实验时候，均能得到 10 个标签为 0 和标签为 1 的有效训练数据。它们数量相等，不会出现直接将所有实验数据做同样处理后出现的不平衡情况。

另外，除有效字符所在位置外，余下的 10 个行和列在 5 个实验中的取法共有 $C_{10}^2 C_8^2 \cdots C_4^2$ 种方法，并且不同的字符进行不同取法进行组合，会产生大量不同的数据集。为了消除不同取法所带来的随机影响，每次采用不同取法整合出标签 0 的数据，与标签为 1 的数据合成一个有效训练数据训练模型，将每种取法均实验一次显然是不可行，也是没有必要的。我们采取下列这样方法：将实验字符所在行或者列外的其他 10 个行或列随机重排两次，每次依次从排列的行或列中选择 2 个，这两个行和列的数据按顺序采集于实验 1，2，…。比如 1 号被试者在进行字符‘B’实验时，将‘B’所在行或列去除，剩余 10 个行或列的一次随机排列为：

9，5，3，7，2，4，12，10，11

这些行或者列闪烁后采集数据标签一定为 0，其中列 9 行 5 闪烁时的数据采集于实验 1，行 3 列 7 采集于实验 2，以此类推。这样的取法每个被试者每个数据采集 2 次。所以每个被试者均可以得到两组有效训练数据。

4.3.2 实验结果

基于 Python3.8 搭建出随机森林字符预测模型，使用上述的数据处理得到训练数据训练模型，对测试集数据做同样的预处理后，送入训练出的模型进行预测计算。最为理想的数据预测情况应该是每个被试者的每个字符，行和列均出现一个 P300 信号，取出行和列，即可定位出预测字符。但模型不可能具有 100%精确度，有可能在其他位置也预测出 P300 信号，我们采用“少数服从多数”的投票方法进行结果的提取。每个字符的 5 组测试数据中，模型预测出出现 P300 信号位置，就给该位置加 1，即投给该位置一票，5 组预测实验结束后，分别取得票最高的行和列作为最终结果，并且以此定位字符作为预测的最终结果。五个被试者的“投票”表格情况如下：

表 4-1 被测者 S1 P300 信号预测位置统计表

char 行列	13	14	15	16	17	18	19	20	21	22
1	0	3	1	0	0	0	1	0	3	1
2	0	1	0	0	5	0	2	0	0	0
3	2	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	2	0	2	1	0
5	0	0	0	1	3	0	0	0	0	1
6	0	0	2	0	0	0	0	0	0	2
7	1	0	2	0	0	0	1	0	1	0
8	0	0	0	0	0	1	0	0	0	0
9	2	1	0	0	3	0	0	1	0	0
10	0	0	1	1	1	0	0	0	0	0
11	0	0	1	0	2	0	3	0	0	0
12	0	2	0	0	0	0	0	2	0	1

表 4-2 被测者 S2 P300 信号预测位置统计表

Char 行列	13	14	15	16	17	18	19	20	21	22
1	0	3	1	0	0	0	1	1	3	0
2	0	1	0	0	4	0	2	0	0	0
3	2	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	2	0	2	1	0
5	0	0	0	2	3	0	0	0	0	1
6	0	0	2	0	0	0	0	0	0	2
7	2	0	2	0	0	0	1	0	2	0
8	0	0	0	0	0	2	0	0	0	0
9	0	2	0	0	3	0	1	1	0	0
10	0	0	1	1	0	1	1	1	0	0
11	1	0	1	0	0	0	3	0	1	0
12	0	3	0	0	0	0	0	2	0	1

表 4-3 被测者 S3 P300 信号预测位置统计表

Char 行列	13	14	15	16	17	18	19	20	21	22
1	0	2	1	0	0	0	0	1	2	0
2	0	0	0	1	4	1	2	0	1	1
3	3	0	0	0	0	0	0	0	0	0
4	0	1	0	0	0	2	0	2	0	0
5	0	0	0	2	3	0	0	0	0	1
6	1	0	3	0	0	0	0	0	0	2
7	2	0	2	0	0	0	1	0	2	0
8	0	0	0	0	0	2	0	0	0	0
9	0	0	0	0	3	0	1	1	1	0

10	0	1	3	1	1	1	1	0	0	0
11	0	0	1	0	0	0	2	0	0	0
12	0	3	0	0	0	0	0	2	0	1

表 4-4 被测者 S4 P300 信号预测位置统计表

行列 \ Char	13	14	15	16	17	18	19	20	21	22
1	0	2	0	0	0	0	0	0	3	0
2	0	1	0	1	4	0	2	0	0	0
3	3	1	0	0	1	0	0	0	0	0
4	0	0	0	0	0	3	0	1	1	1
5	0	0	0	2	2	0	1	0	0	1
6	0	0	2	0	0	0	0	0	0	2
7	2	0	2	0	0	0	1	0	2	0
8	0	1	0	1	0	2	0	0	0	0
9	1	0	0	0	2	0	0	1	1	1
10	0	0	1	2	0	1	1	0	0	0
11	0	0	0	0	0	0	2	0	0	0
12	0	2	0	0	0	0	0	2	0	2

表 4-5 被测者 S5 P300 信号预测位置统计表

行列 \ Char	13	14	15	16	17	18	19	20	21	22
1	0	3	1	0	0	0	1	1	2	0
2	0	1	0	0	3	0	2	0	0	0
3	1	0	0	1	1	1	0	0	0	0
4	0	0	0	0	1	2	0	2	1	0
5	0	1	1	2	2	0	0	0	0	1
6	0	0	2	0	0	0	0	0	0	2
7	2	0	3	0	0	1	1	0	2	0
8	0	0	0	0	0	2	0	0	0	0
9	0	2	0	0	2	0	0	1	1	1
10	0	1	1	2	1	0	1	0	0	0
11	1	0	0	0	0	0	2	0	1	0
12	0	3	0	0	0	0	0	2	0	2

依据每个字符的 5 轮实验中，模型预测的各位置出现 P300 信号次数，分别取出现次数最高的行和列序号，以此定位出需要识别的字符，其结果如下：

表 4-6 待识别字符统计表

字符 \ 被试者	S1	S2	S3	S4	S5
Char13	M	O	M	M	M
Char14	F	F	F	F	F
Char15	5	5	8	5	5

Char16	2	2	2	2	2
Char17	I	I	I	I	I
Char18	T	T	T	T	T
Char19	K	K	K	K	K
Char20	X	X	X	X	X
Char21	A	A	A	A	A
Char22	0	/	/	0	0

在给出的五个测试结果中，随机森林算法对每个受试者的识别准确度情况如下表：

表 4-7 随机森林算法前五个测试数据精度表

受试者	S1	S2	S3	S4	S5
识别准确度	100%	80%	80%	100%	100%

4.3.3 简单的对比实验

为了检验随机森林算法的优越性，我们使用同样的数据训练最经典，最简单的机器学习模型 KNN,采用同样“少数服从多数”方法，模型的字符预测结果如下：

表 4-8 KNN 模型字符预测表

被试者 / 字符	S1	S2	S3	S4	S5
Char13	M	N	M	O	M
Char14	F	F	F	F	F
Char15	Y	5	5	5	5
Char16	2	2	2	2	2
Char17	I	J	I	I	I
Char18	T	T	K	T	T
Char19	K	K	K	K	K
Char20	X	X	X	X	X
Char21	A	A	A	A	A
Char22	L	/	/	0	0

在给出的五个测试结果中，KNN 算法对每个受试者的识别准确度情况如下表：

表 4-9 KNN 算法前五个测试数据精度表

受试者	S1	S2	S3	S4	S5
识别准确度	80%	60%	100%	80%	100%

与随机森林法对比，KNN 在应对高维二分类数据时性能较低，但其大部分结果与随机森林法相同。与题目提供的前五个字符的识别结果相比，随机森林法仅有两个结果预测错误，正确率高达 92%，而 KNN 算法正确率仅为 80%。在算法运行时间上看，KNN 是明显优于随机森林的。二者的正确率和运行时间如下表所示：

表 4-10 随机森林与 KNN 对比表

算 法	KNN	随机森林
时 间（s）	0.0529	0.1730
精 度（%）	80	92

5．问题二的模型建立与求解

5.1 问题二的分析

题目给出的原始的 20 个通道采集的脑电数据非常庞大，其中参杂着很多冗余信息，不仅会带来系统的复杂性，增大系统数据传输和处理的压力，甚至会造成模型的分类和识别精度的下降，因此我们将剔除影响较小甚至没有影响的通道以选取最优的通道组合。传统的通道选择方法分为两类，一类是经验选择法，另外一种是机器学习法。经验选择法需要扎实的生物学相关的知识和前期大量的实验来论证各个通道的比例权重，而且其选择的算法不能满足每个被试者的特点，会造成重要数据丢失，影响实验结果。机器学习算法则是通过挖掘数据内在的特点，能够根据不同的被试者自动调整各个通道的采集数据的重要性。所以我们选择了机器学习算法来解决通道选择问题。

考虑到脑电信号子集的空间稀疏性，目前已经存在多种稀疏贝叶斯回归方法，比如对所有脑电通道施加L_1范数约束，选择出具有空间稀疏性的最优通道子集 LASSO 方法[8]，对通道不同组合施加L_1范数约束的组 GROUP-LASSO 法[9]。本文以“spike and slab”先验建立稀疏贝叶斯回归模型，以此训练出每个被试者的最佳通道组合。

5.2 模型建立

不失一般性，我们以第一个被试者为例来推导我们的模型。首先，将某个采样点 20 个通道采集的数据记作d_i，d_i为一个长度为 20 的向量，即为给的表格数据中的一行。将每行或每列闪烁时刻起N个采样数据作为有效数据，每个被试者均进行 5 轮实验，每轮实验行和列一共闪烁 12 次，所有每个被试者有$60N$有效数据。和问题一保持一致，仍然选取$N=125$。将被试者所有的有效数据组合成一个矩阵，并且用字母D表示，$D=(d_1,d_2,...,d_{60k})^T$. D是维度为7500×20的矩阵, 在压缩感知领域也称为“字典”。同样地，我们也将 P300 信号进行标签化处理，记作y，y是维度为 7500 的向量。如果相应的采样时刻出现 P300 信号，那么y对应的位置为 1，否则为 0。20 个通道的权重向量为x。那么权系数求解问题就转换成一个线性回归模型：

$$y=Dx+\varepsilon \tag{7}$$

为了方便模型的推导，将字典D的维度记作$P\times K$. ε为具有精度为σ的高斯零均值的随机变量。为了得到关于x的稀疏解，我们引入“spike and slab”稀疏促进函数[5]。

$$x|z\sim\prod_{j=1}^{K}[(1-z_j)\delta(x_j)+z_jN(x_j|0,\tau_0)] \tag{8}$$

其中x_i是K维权重系数向量x的第i个分量, 他们是K个满足均值为 0，方差为τ_0且互相独立高斯变量。z称为支撑向量，是K个$i.i.d$伯努利分布的二维随机变量。$z_i=1$意味着$x_i\neq0$，进一步意味着通道 i 是需要选择的，是对睡眠状态判定有着至关重要的通道；$z_i=0$意味着$x_i=0$则意味这条通道是多余的。$\delta(\cdot)$代表冲激函数。支撑向量满足的分布为

$$z|\pi \sim \prod_{j=1}^{K} Ber(\pi_j) \tag{9}$$

根据贝叶斯准则，隐藏变量x,z的后验概率分布为

$$p(x,z|y) = \frac{p(y|x)\,p(x|z)\,p(z)}{\int p(y|x)\,p(x|z)\,p(z)\,dxdz} \tag{10}$$

上式分母为一个归一化常数，不影响后验分布的具体形式。观察上式，分子为K维高斯分布与K维伯努利分布以及K维“spike and slab”分布的乘积形式，想要得到其准确的闭式数学表达式是不可能的，进一步分母是分子的复杂的多元积分，分子闭式表示式的难以求解必然导致计算积分值也同样是不可行的，所以式（10）是无法求得解析解的，即无法精确计算权重系数以及其支撑向量的精确后验分布。但在实际工程问题中，可以用来得到上式的近似解。常见的此类近似算法有 MCMC，Gibbs sampler，VB 等等，但上述算法均存在效率低，收敛情况难以检测等问题。本题中，我们采取期望传播算法解决上式（Expectation Propagation, 简称 EP 算法）。EP 算法高效，并且收敛情况易于检测优点是我们选择的主要原因[4]。同样通过贝叶斯准则得到参数的后验分布天生具有防止过拟合的优点，这点是最大似然估计和最大后验准则所不具备的。

5.3 模型的求解

Expectation Propagation, 简称 EP 算法是一种稳定，快速的分布近似算法。它的核心思想是在指数分布族（Exponential Family）概率分布中选取与似然函数耦合的分布，最小化两者的 KL 散度，从而求得模型的近似分布的解析表达式。将式（7）~（9）带入（10）得到权重系数及其支撑变量的后验分布表达式为：

$$\begin{aligned} p(x,z|y) &= \frac{1}{Z}\,p(y|x)\,p(x|z)\,p(z) \\ &= \frac{1}{Z}N(y|Dx,\sigma_0^2 I)\prod_{j=1}^{K}((1-z_j)\delta(x_j) + z_j N(x_j|0,\tau_0))\,p(z) \end{aligned} \tag{11}$$

注意到似然函数$p(y|x)$为多维高斯分布，并且支撑集满足伯努利分布，而高斯-伯努利分布本身就是耦合的分布，由此可以推断出x,z的联合后验分布也一定是高斯-伯努利分布的。为了避免整体近似的复杂度，进一步观察上式，其可以分解为三项

$$\begin{aligned} f_1(x) &= N(y|Dx,\sigma_0^2 I) \\ f_2(x,z) &= \prod_{j=1}^{K}((1-z_j)\delta(x_j) + z_j N(x_j|0,\tau_0)) = \prod_{j=1}^{K} f_{2,j}(x_j,z_j) \\ f_3(\boldsymbol{z}_i) &= \prod_{j=1}^{K} Ber(\pi_j) \end{aligned} \tag{12}$$

第二项又可以进一步分解为K项，可以使用 EP 算法分别推导上三式的近似分布从而得到整体的近似解。EP 算法的第一步就是选择合适的近似因子，近似因子的选择一方面要保证简单性，另外一方面又要保证模型具有足够的复杂度以能够很好的拟合数据内在的分布。注意到f_1只与x有关，f_2与x,z有关，f_3与z有关。针对本模型，我们选择 3 个属于指数分布族的概率分布形式如下：

$$
\begin{aligned}
q_1(x) &= N(\boldsymbol{x}_i|\boldsymbol{m}_1,\boldsymbol{\Sigma}_1)\\
q_2(x,z) &= N(x|\boldsymbol{m}_2,\boldsymbol{\Sigma}_2)\prod_{j=1}^{K}Ber(z_j|\mu_{1j}) = \prod_{j=1}^{K}N(x_j|m_{2j},\Sigma_{2j})Ber(z_j|\mu_{1j})\\
q_3(\boldsymbol{z}_i) &= \prod_{j=1}^{K}Ber(z_j|\mu_{2j})
\end{aligned}
\tag{13}
$$

三者相乘可以得到整体后验近似分布为

$$
\begin{aligned}
Q(x,z) &= N(x|\boldsymbol{m},\boldsymbol{\Sigma})\prod_{j=1}^{K}Ber(z_j|\mu_j)\\
\boldsymbol{\Sigma}^{-1} &= (\boldsymbol{\Sigma}_1^{-1}+\boldsymbol{\Sigma}_2^{-1})^{-1}\\
\boldsymbol{m} &= \boldsymbol{\Sigma}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{m}_1+\boldsymbol{\Sigma}_2^{-1}\boldsymbol{m}_2)\\
\mu_j &= \left(\frac{(1-\mu_{1j})(1-\mu_{2j})}{\mu_{1j}\mu_{2j}}+1\right)^{-1}
\end{aligned}
\tag{14}
$$

观察f_1,f_3以及q_1,q_3可以发现，$q_1(x),q_3(z)$的分布参数都是可以通过先验概率精确计算的，仅仅q_2需要使用 EP 算法求解。通过二次项匹配，可以计算q_1分布的参数为：

$$
\begin{aligned}
\boldsymbol{\Sigma}_1^{-1} &= -\frac{1}{\sigma_0^2}D^TD\\
\boldsymbol{\Sigma}_1^{-1}\boldsymbol{m}_1 &= \frac{1}{\sigma_0^2}D^Ty
\end{aligned}
\tag{15}
$$

q_3的参数分布则与先验分布相等即：

$$
\mu_{2j} = \pi_j \tag{16}
$$

同样使用 EP 算法分别优化q_2的每一项q_{2j}，即可得到q_2整体后验分布。考虑到更新q_{2j}，第一步就是从整体后验分布中去除掉这一项，计算其 cavity 分布

$$
Q^{\backslash 2j}(x,z) = \frac{Q(x,z)}{N(x_j|m_{2j},\Sigma_{2j})Ber(z_j|\mu_{1j})} = \frac{N(x|\boldsymbol{m},\boldsymbol{\Sigma})\prod_{j=1}^{K}Ber(z_j|\mu_j)}{N(x_j|m_{2j},\Sigma_{2j})Ber(z_j|\mu_{1j})} \tag{17}
$$

通过边缘积分，得到x_j,z_j的联合边缘 cavity 分布为

$$
p(x_j,z_j) = N(x_j|m_j^{\backslash 2j},\Sigma_j^{\backslash 2j})Ber(z_j|\mu_j^{\backslash 2j}) \tag{18}
$$

其中

$$
\begin{aligned}
\Sigma_j^{\backslash 2j} &= ((\boldsymbol{\Sigma})_{jj}^{-1}-\Sigma_{2j})^{-1}\\
m_j^{\backslash 2j} &= \Sigma_j^{\backslash 2j}(m_j(\boldsymbol{\Sigma})_{jj}^{-1}-m_{2j}\Sigma_{2j}^{-1})\\
\mu_j^{\backslash 2j} &= \left(\frac{(1-\mu_j)\mu_{1j}}{(1-\mu_{1j})\mu_j}+1\right)^{-1}
\end{aligned}
\tag{19}
$$

然后最小化Q^{new}与$f_{2j}Q^{\backslash new}$之间的 KL 散度。因为分布Q属于指数分布族分布，最小化 KL 散度等价于实现二者之间的矩量匹配。$f_{2j}Q^{\backslash new}$的归一化常数为

$$
\begin{aligned}
Z_{2j} = Q^{\backslash 2j}(x,z)f_{2.j}(x_j,z_j) &= \sum_{\boldsymbol{z}_i}\int Q^{\backslash 2j}(x,z)\left((1-z_j)\delta(x_j)+z_jN(x_j|0,\tau_0)\right)dx\\
&= \mu_j^{\backslash 2j}N(0|m_j^{\backslash 2j},\Sigma_j^{\backslash 2j}+\tau_0)+(1-\mu_j^{\backslash 2j})N(0|m_j^{\backslash 2j},\Sigma_j^{\backslash 2j})
\end{aligned}
\tag{20}
$$

二者之间的矩量匹配经过推导得出：

$$\begin{aligned} m_j^{new} &= E_{\frac{1}{Z_{2j}}Q^{\backslash 2j}(x,z)f_{2\cdot j}(x_j,z_j)}(x_j) = \frac{1}{Z_{2j}}\sum_{z_j}\int x_j\, p(x_j,z_j) f_{2\cdot j}(x_j,z_j)dx_j \\ &= \frac{1}{Z_{2j}}\mu_j^{\backslash 2j} N(0\,|m_j^{\backslash 2j}, \Sigma_j^{\backslash 2j}+\tau_0)\frac{\tau_0 m_j^{\backslash 2j}}{\Sigma_j^{\backslash 2j}+\tau_0} \end{aligned} \tag{21}$$

$$\begin{aligned} m_j^{new2} + \Sigma_j^{new} &= E_{\frac{1}{Z_{2j}}Q^{\backslash 2j}(x,z)f_{2\cdot j}(x_j,z_j)}({x_j}^2) = \frac{1}{Z_{2j}}\sum_{z_j}\int {x_j}^2 p(x_j,z_j) f_{2\cdot j}(x_j,z_j)dx_j \\ &= \frac{1}{Z_{2j}}\mu_j^{\backslash 2j} N(0\,|m_j^{\backslash 2j}, \Sigma_j^{\backslash 2j}+\tau_0)\left[\left(\frac{\tau_0 m_j^{\backslash 2j}}{\Sigma_j^{\backslash 2j}+\tau_0}\right)^2 + \frac{\tau_0\Sigma_j^{\backslash 2j}}{\Sigma_j^{\backslash 2j}+\tau_0}\right] \\ \Longrightarrow \Sigma_{ij}^{new} &= \frac{1}{Z_{2j}}\mu_{ij}^{\backslash 2j} N(0\,|m_j^{\backslash 2j}, \Sigma_j^{\backslash 2j}+\tau_0)\left[\left(\frac{\tau_0 m_j^{\backslash 2j}}{\Sigma_{ij}^{\backslash 2j}+\tau_0}\right)^2 + \frac{\tau_0\Sigma_j^{\backslash 2j}}{\Sigma_j^{\backslash 2j}+\tau_0}\right] \\ &\quad - \left(\frac{1}{Z_{2j}}\mu_j^{\backslash 2j} N(0\,|m_j^{\backslash 2j}, \Sigma_j^{\backslash 2j}+\tau_0)\frac{\tau_0 m_j^{\backslash 2j}}{\Sigma_j^{\backslash 2j}+\tau_0}\right)^2 \end{aligned} \tag{22}$$

$$\mu_j^{new} = E_{\frac{1}{Z_{2j}}Q^{\backslash 2j}(x,z)f_{2\cdot j}(x_j,z)}(z_j) = \mu_j^{\backslash 2j}\frac{1}{Z_{2j}} N(0\,|m_j^{\backslash 2j}, \Sigma_j^{\backslash 2j}+\tau_0) \tag{23}$$

最后，我们更新f_{2j}的参数：

$$\begin{aligned} {m_{2j}}^{new} &= \Sigma_{2j}^{new}(\Sigma_j^{new-1} m_j^{new} - \Sigma_j^{\backslash 2j-1} m_j^{\backslash 2j}) \\ {\Sigma_{2j}}^{new} &= (\Sigma_j^{new-1} - \Sigma_j^{\backslash 2j-1})^{-1} \\ {\mu_{1j}}^{new} &= \left(\frac{(1-{\mu_j}^{new}){\mu_j}^{\backslash 2j}}{(1-{\mu_j}^{\backslash 2j}){\mu_j}^{new}} + 1\right)^{-1} \end{aligned} \tag{24}$$

根据 EP 算法计算出的x,z的后验分布并且支撑向量z表示着各个通道不等于 0 的概率，在模型收敛以后，我们按照$z_i=1$的概率大小排序，依次选择最重要的通道。完整的算法如下所示：

*Step*1: 初始化近似项$q_a, a=1,2,3$

*Step*2: 重复以下步骤直到收敛：

对每个q_{ij}:

计算*Cavity* 分布：$Q^{\backslash 2j} \propto \frac{Q}{q_{2j}}$

最小化：$KL(f_{2j}Q^{\backslash 2j} \| Q_{new})\, w.r.t. Q_{new}$

计算：$q_{2j} \propto \frac{Q_{new}}{Q^{\backslash 2j}}$ 以更新参数$m_{2j}, \Sigma_{2j}, \mu_{2j}$

计算联合近似分布以更新参数$\boldsymbol{\Sigma}, \boldsymbol{m}, \mu_j$

*Step*3: 根据μ_j大小排序，选择最重要的通道

5.4 基于 EP 算法的贝叶斯回归学习算法

按照 5.1 节描述的方法建立字典D后，考虑到一些高频噪声对实验数据的扰动，我们设计通带为 0.1Hz 到 20Hz, 阻带增益-80dB 的 46 阶带通巴特沃斯 IIR 滤波器，滤除因为仪器噪声以及如眨眼等动作带来的干扰。我们使用 python3.8 搭建提出的算法，将五个被试者的数据送入基于 EP 的“spike and slab”稀释贝叶斯回归模型进行训练，五个被试者的

支撑向量z的后验分布不等于 0 的概率即μ，如下图所示。其中横坐标为通道标号，纵坐标为通道非 0 的概率。

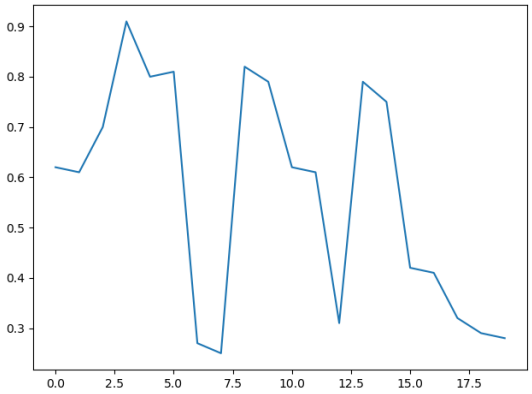


图 5-1 S1 支撑向量的后验分布图

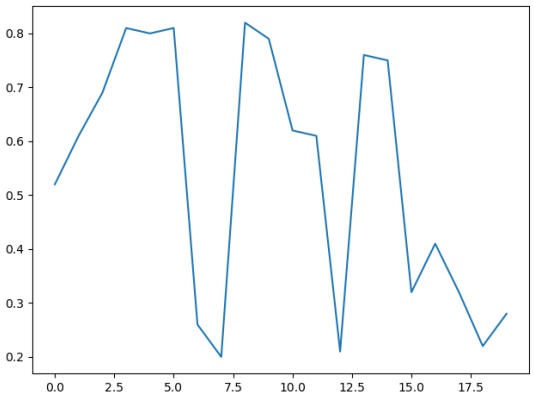


图 5-2 S2 支撑向量的后验分布图

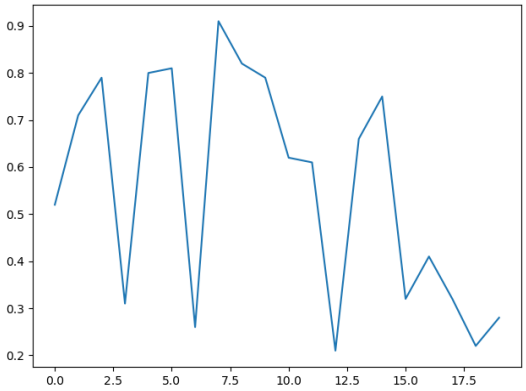


图 5-3 S3 支撑向量的后验分布图

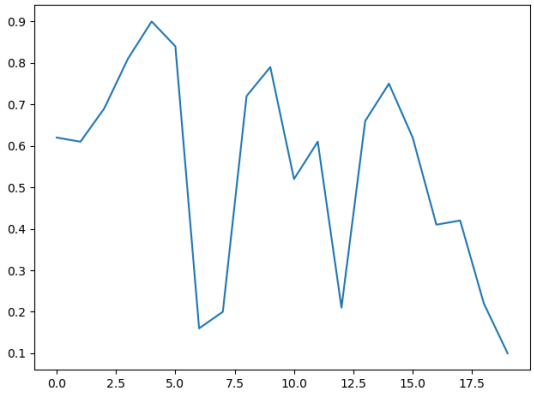


图 5-4 S4 支撑向量的后验分布图

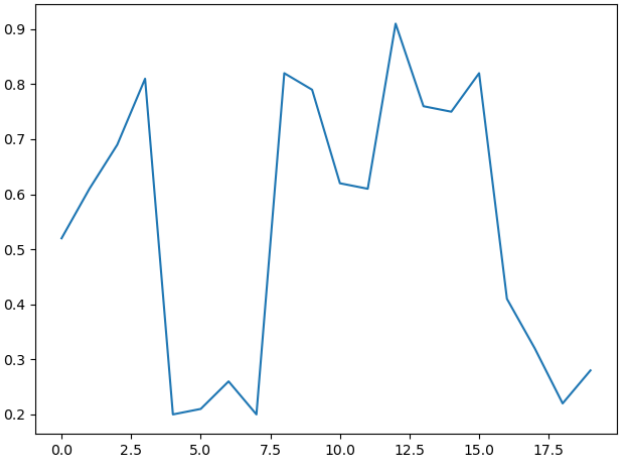


图 5-5 S5 支撑向量的后验分布图

将算法学习的支撑区z的后验分布，按照非 0 概率按照大小顺序排列，取最前面的若干个通道，得到五个被试者的通道选择结果如下表所示，其中✔表示此通道重要，✘表示此通道对被试者而言可以忽略。

表 5-1 S1-S5 的通道选择结果表

被试者 / 通道	S1	S2	S3	S4	S5
1	✔	✔	✔	✔	✔
2	✔	✔	✔	✔	✔
3	✔	✔	✔	✔	✔
4	✔	✔	✕	✔	✔
5	✔	✔	✔	✔	✕
6	✔	✔	✔	✔	✕
7	✕	✕	✕	✕	✕
8	✕	✕	✔	✕	✕
9	✔	✔	✔	✔	✔
10	✔	✔	✔	✔	✔
11	✔	✔	✔	✔	✔
12	✔	✔	✔	✔	✔
13	✕	✕	✕	✕	✔
14	✔	✔	✔	✔	✔
15	✔	✔	✔	✔	✔
16	✔	✕	✔	✔	✔
17	✕	✕	✕	✕	✕
18	✕	✕	✕	✔	✕
19	✕	✕	✕	✕	✕
20	✕	✕	✕	✕	✕

题目要求根据每个被试者的通道选择情况，确定一个对所有被试者的最优通道选择，我们采用一种“少数服从多数“原则，即一个通道如果被 3 个及其以上被试者选择，即认为该通道应该被选入最优通道，否则拒绝该通道，结合表(5-1），我们给出的最优通道组合为 1，2，3，4，5，6，9，10，11，12，14，15，16 组成的 13 个通道，剔除通道 7，8，

13，17，18，19，20 一共 7 个通道。

基于选择的通道组合，剔除掉不重要的通道采集的数据后，将有效数据依照问题一提出的数据预处理方法处理数据，然后采用随机森林算法训练字符识别模型，将 5 个被试者需要识别的字符数据送入训练好的模型进行预测，S1，S4，S5 被试者 char13-char22 以及 S2,S3 被试者 char13-char21 预测结果如下：

表 5-2 五位被试者的目标字符预测结果表

被试者 / 字符	S1	S2	S3	S4	S5
Char13	M	M	M	M	M
Char14	F	L	F	F	F
Char15	5	5	8	5	0
Char16	2	2	2	2	2
Char17	I	I	I	I	I
Char18	T	T	N	T	T
Char19	K	K	K	K	K
Char20	X	X	X	X	X
Char21	A	A	A	A	A
Char22	0	/	/	0	0

采用选定的最优通道组合，在给出的五个测试结果中，随机森林算法对每个受试者的识别准确度情况如下表：

表 5-3 KNN 算法前五个测试数据精度表

受试者	S1	S2	S3	S4	S5
识别准确度	100%	80%	80%	100%	80%

采用依据最优组合通道采集数据，训练模型并预测字符结果，与问题一中采用所有通道数据，训练相同模型并预测的字符结果相比.可以看出，虽然舍弃了通道 7，8，13，17，18，19，20 共 7 个通道，模型预测结果依旧具有很好的稳定性，预测结果仅仅 5 不同。尤其注意到，与问题中给出的 char13-char17 的结果相比，仅有 2 个是不符的，其余均可以准确识别出，模型预测精确达到 92%。因此基于“spike and slab”稀疏先验的贝叶斯回归模型，使用 EP 算法解出的通道后验支撑权重，能够很好符合实际通道权重分布。依据通道权重选择出的最优通道组合，不仅大大减少了数据冗余，噪声数据，依旧保持了模型的预测精度。

6．问题三的模型建立与求解

6.1 问题三的分析

在 P300 脑-机接口系统中，需要花费很长时间来获取有标签样本，而大量的无标签样本却很容易得到，该问题就对应一个典型的半监督学习(Semi-supervised Learning)问题。半监督学习机制的原理可形象的由下图说明：

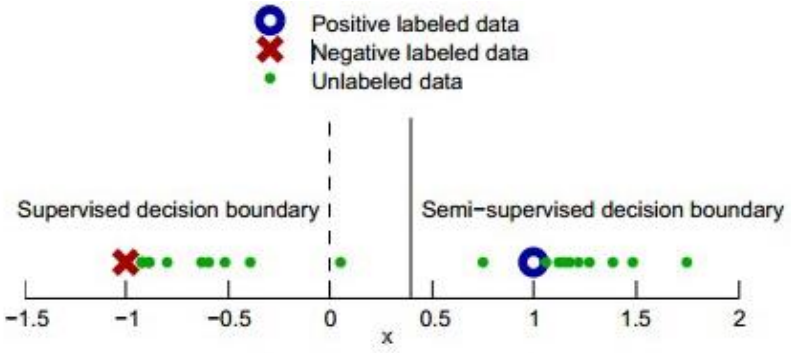


图 6-1 半监督学习二分类示意图

当训练集中只存在少量的有标记样本，而其余样本均为无标记时，仅用有标记样本集训练分类器得到的效果往往受到训练样本数量的限制，而加入无标记样本进行训练可以很好的提高分类的准确率[1]。上图中只有两个有标记的样本 X、O，剩下的绿点是无标记样本，通过无标记样本的加入，原来的分类界限从 0 移到了 0.5 处，更好地拟合了样本现实分布。

本题中我们可根据半监督学习的原理将第二问中所选 13 个最优通道训练集脑电信号进行预处理后分成两类：有标签样本$Lable=\{(x_i,y_i)\}(i=1,...l)$和无标签样本$Unlabled=\{(x_i)\}(i=l+1,..l+u)$。其中有标签样本集大小为$l$，无标签样本集大小为$u$，一般而言，半监督学习侧重于在有监督分类中加入无标记样本，增强分类的效果，因此在分类样本集时应满足有标签样本集大小远小于无标签样本集的大小，即$l \ll u$[2]。并保证有标签样本集中正负样本的比例应尽量与真实的先验分布相同

6.2 模型的建立

6.2.1 半监督学习支持向量机（S3VM）方法

半监督学习支持向量机（S3VM）将已知的支持向量机分类器扩展到半监督场景[3]。除了使用训练集中的标记部分来最大化类之间的边界，分类器还通过迫使决策函数遍历低密度区域来最大化利用未标记的数据。这种方法采用了群体假设，即相互接近的样本可能具有相同的标签。

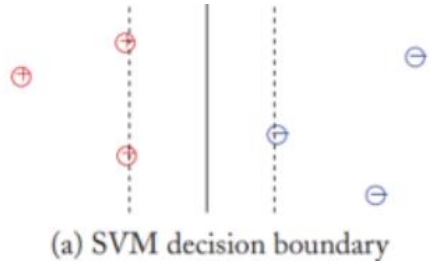


图 6-2 传统的 SVM 决策边界

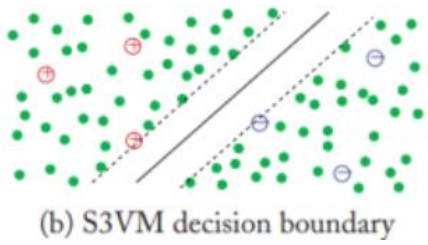


图 6-3 半监督 S3VM 决策边界

在 S3VM 示意图中，空心图形代表有标签的数据，而实心图形代表无标识数据。新的决策边界可以很好地将无标签数据分成两类，并且也正确分类了有标签数据。和支持向量机的组成原理类似，分类面的生成一样使用了最大-最小理论。

在本题的应用场景下，我们考虑线性二元分类问题，在标记样本$Lable=\{(x_i,y_i)\}_{i=1}^{l}$和未标记样本$Unlabled=\{(x_i)\}_{i=l+1}^{n}$中，$x_i$是$d(d=N*k)$维向量，N 为 300ms 内采样点的个数，k 为选取的 k 个最优通道，$y_i\epsilon\{-1,1\}$表示所选样本内是否发生 P300 信号。在处理 S3VM 时，需要优化的目标函数取决于分离超平面参数(w,b)和未知标签$y_{i=l+1}^{n}$

$$P(w,b,y_{i=l+1}^{n}) := \frac{1}{2}\|w\|^2+C\sum_{i=1}^{l}V(y_i,a_i)+C^*\sum_{i=l+1}^{n}V(y_i,a_i) \tag{25}$$

其中$a_i=w^Tx_i+b$是线性预测，Hinge loss 为$V(y_i,a_i):=\max\{0,1-a_iy_i\}$。超参数 C 和 C*分别表示标记和未标记错误项的权重。由于标记样本的数量远小于未标记样本，为了避免所有示例都被归属于同一类的琐碎解决方案，通过指定未标记样本中正负样本的期望比率 r 来进行约束（在该实验中 r 为 1/5）：

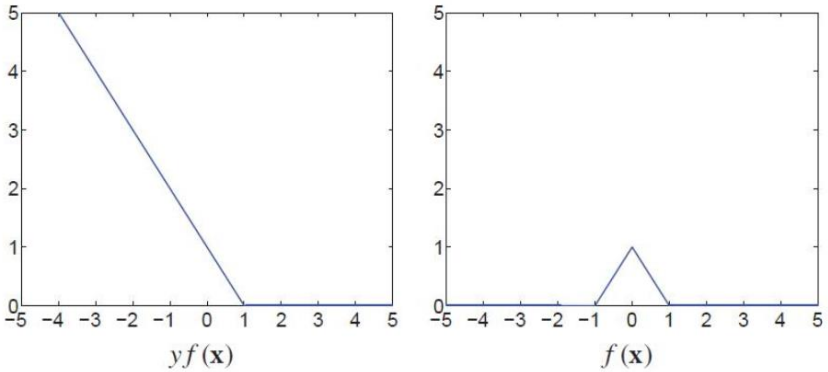
$$\frac{1}{u}\sum_{i=l+1}^{n}\max\{y_i,0\}=r$$

该式等价于

$$\frac{1}{u}\sum_{i=l+1}^{n}y_i=2r-1 \tag{26}$$

式（25）的优化技术遵循两种广泛的策略，连续策略和组合策略。前一种方法用无标签样本的预测值$\hat{y}_i=sign(f(x_i))=sign(w^Tx_i+b)$替换$y_{i=l+1}^{n}$，则该样本上的 hinge 损失函数替换为 hat loss：

$$\begin{aligned} c\left(x,\hat{y},f(x)\right) &= \max\left(1-\hat{y}(w^Tx+b),0\right) \\ &= \max(1-sign(w^Tx+b)(w^Tx+b),0) \\ &= \max(1-|w^Tx+b|,0) \end{aligned} \tag{27}$$



(a) the hinge loss　　(b) the hat loss

图 6-4 两类损失函数图

得到的函数是一个只依赖于(w,b)的非凸函数，同时获得非线性平衡约束，解决这一问题的常见方法是处理约束的线性松弛，但由于凸性的损失，现成的 off-the-shelf dual-based SVM 软件不能直接用于优化（25）。

根据组合的策略可以求解（25），一但未知标签$y_{i=l+1}^{n}$确定，可以得到一个标准的 SVM 公式：

$$I(y_{i=l+1}^{n}):=\min_{w,b}P(w,b,y_{i=l+1}^{n}) \tag{28}$$

组合方法的目的是在二进制变量$y_{i=l+1}^{n}$的基础上最小化$I(y_{i=l+1}^{n})$，这也适用于非线性情况。

6.2.2 模型的求解

为了简单起见，用线性公式来阐述算法思想，容易推广到非线性情况中。首先，用训练集中的标记部分初始化决策函数。在每次迭代中，通过考虑当前决策函数和平衡约束，给训练集中未标记的样本$\{x_i\}_{i=l+1}^{n}$通过拉格朗日方法贴上标签。内部支持向量机在扩展的标记集上再次训练，包括$\{x_i\}_{i=1}^{l}$和通过拉格朗日启发式标记处理的样本。在每次迭代中增加权重 C*，直到它被分配到与 C 相同的值，这是一种常见的基于退火的全局优化方法，其目的是随着学习过程的发展，越来越重视初始为未标记的样本。

拉格朗日启发式的工作方式为，一旦(w,b)确定，（25）由式中的可变部分决定：

$$U(y_{i=l+1}^{n}):=\sum_{i=l+1}^{n}\max\{0,1-y_i(w^Tx_i+b)\} \tag{29}$$

加上平衡约束条件（4），便归结为一个约束最优标记问题：

$$\begin{aligned}&\min_{y_{i=l+1}^{n}}U(y_{i=l+1}^{n})\\&\frac{1}{u}\sum_{i=l+1}^{n}y_i=2r-1\end{aligned} \tag{30}$$

利用拉格朗日乘子λ放松约束，求解相应的对偶问题

$$\begin{aligned}&\max_{\lambda}\ \min_{y_{i=l+1}^{n}}\ L(\lambda,y_{i=l+1}^{n})\\&:=\max_{\lambda}\ \min_{y_{i=l+1}^{n}}\ \sum_{i=l+1}^{n}\max\{0,1-a_iy_i\}+\lambda\left(\sum_{i=l+1}^{n}y_i-\beta\right)\end{aligned} \tag{31}$$

其中a_i是预测值w^Tx_i+b,常数项β定义为$(2r-1)u$,上式等价于：

$$L(\lambda,y_{i=l+1}^{n})=\sum_{i=l+1}^{n}\max\{\lambda y_i,1-a_iy_i+\lambda y_i\}-\lambda\beta \tag{32}$$

定义$F(\lambda,y)=\sum_{i=l+1}^{n}\max\{\lambda y_i,1-a_iy_i+\lambda y_i\}$。若将拉格朗日乘子固定为起始值$\lambda_0$，上述优化问题在 y 变量中可分离，每个分量$y^*:=arg\min_y F(\lambda_0,y)$可以独立计算：

$$y_i^*=arg\ \min_{y_i\in\{-1,1\}}\ \max\{0,1-a_iy_i\}+\lambda_0y_i,\ i=l+1,...n \tag{33}$$

在（9）中插入y^*并更新乘法值，所得$L(\lambda)=\min_{y_{i=l+1}^{n}}L(\lambda,y_{i=l+1}^{n})$是$\lambda$的一个凹函数，通过该性质获得拉格朗日启发式后续迭代的更新值λ^{next}。这种迭代方法称为切割平面，是一种典型的非可微优化方法，用于求解拉格朗日对偶，让(λ^a,y^a)和(λ^b,y^b)成为一对对偶解：

$$\sum_{i=l+1}^{n} y_i^a - \beta < 0 \ and \ \sum_{i=l+1}^{n} y_i^b - \beta > 0 \tag{34}$$

$$L(\lambda^{next}, y^a) = L(\lambda^{next}, y^b) \tag{35}$$

然后让y^{next}通过在（10）中插入λ^{next}获得标记，根据约束违反的标志 $\sum_{i=l+1}^{n} y_i^{next} - \beta$ 设定$y^a = y^{next}$或$y^b = y^{next}$，并迭代此过程至收敛。

如上所述，当平衡约束满足期望的公差范围时，拉格朗日启发返回标记$y_{i=l+1}^n$，在随后的 S3VM 退火迭代中，考虑增强的标记集，并计算一个新的分离函数。该半监督学习方法的伪代码如下所示：

*Step*1: **输入：**最优 k 个通道采集的有标记脑电数据$\{x_i, y_i\}_{i=1}^{l}$，未标记脑电数据$\{x_i\}_{i=l+1}^{n}$，正负样本比例 r，标准的退火序列 C*

*Step*1: C ← 在标记集$\{x_i, y_i\}_{i=1}^{l}$上交叉验证 SVM

*Step*2: H ← 通过标记集$\{x_i, y_i\}_{i=1}^{l}$支持向量机得到的分离函数

*Step*3: **for** C* 从退火序列中提取 **do**：

*Step*4: |α ← 未标记样本$\{x_i\}_{i=l+1}^{n}$到分离边界 H 的距离

*Step*5: |通过α、r 及式(6)-(13)计算标签$y_{i=l+1}^n$

*Step*6: |H← 用 C*加权初始无标记样本，通过增强的训练集获得 SVM 的分离函数

*Step*7: **end**

*Step*8: **输出：**训练的分类器

下图为拉格朗日-S3VM 的一些选定退火迭代的中间过程。

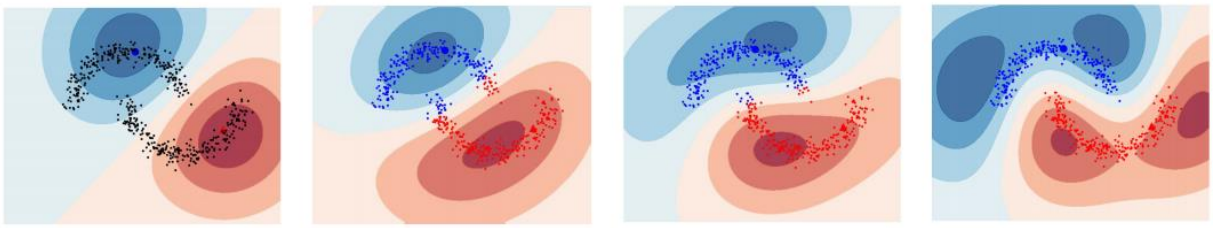


图 6-5 拉格朗日-S3VM 退火迭代过程示意图

图 6-5（a）中蓝色大圆和红色大三角形为两个标记示例，其余黑点为未标记示例，分类得到了边界的表面轮廓。在这种初始化之后，未标记的模式被越来越多地考虑，决策函数在(b)和(c)中演化，直到在(d)中达到平衡的解。

6.3 实验情况

在本实验中，我们选取了 20 个通道中的 13 个通道的信号（Fz, F3, F4, C3, Cz, C4, CP3, CP4, CP5, CP6, P3, P4, P7），每次闪烁截取 125 个点（500ms）的信号作为一个样本，每个样本可以表示为一个 10*75 的向量，并通过一个通带为 0.1 到 20Hz 的巴特沃斯带通滤波器进行滤波。在本实验中，我们选择 12 个字符中 10 个字符的数据作为训练集（共$12 \times 5 \times 10 \times 5 = 3000$个样本），剩下的数据作为验证集（共$12 \times 5 \times 2 \times 5 = 600$个样本）。对训练集，我们将前 30%的作为有标签数据（900 个样本），后 70%的作为无标签数据（2100 个样本），形成标记的数据集$\{x_i, y_i\}_{i=1}^{l}$和无标记的数据集$\{x_i\}_{i=l+1}^{n}$

实验流程如下图所示：

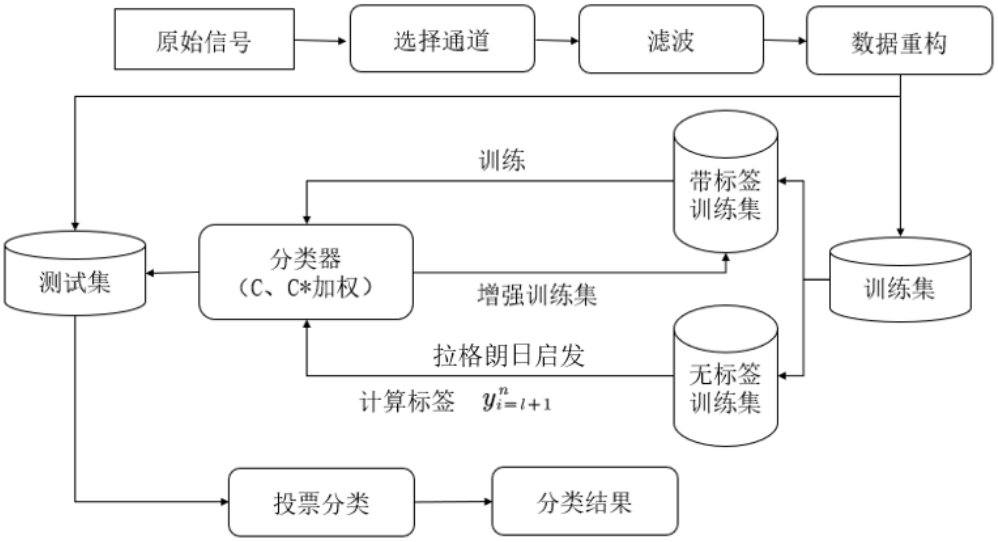


图 6-6 基于半监督判别的 P300 信号分类流程图

在实验中，对算法与所选 S3VM 求解器进行的设置如下：对于内部 SVM 模型，选取高斯核$K(x,x')=\exp(-\gamma\|x-x'\|^2)$，监督求解器通过交叉验证 C 的一组非常小的值来内部预置。C 从$\{2^i, i\epsilon[0,5]\}$中选择，C*从$\{(1/10)C,(1/4)C,(1/2)C,C\}$中取值，从而产生四个退火迭代。

经过半监督学习的分类器在验证集上的平均准确率达到 67.3%。为了验证半监督学习得到的分类器的性能，我们将学习得到的模型用于测试数据(char13-char17)，结果如下：

表 6-1 S3VM 模型预测结果表

Char13	Char14	Char15	Char16	Char17
M(A)	F(L)	5(Y)	2	I

可见在大量未标记样本和少数标记样本下的半监督学习已经拥有较高的识别准确度，并且由于不再需要对样本集进行人工标记，节省了大量的时间。

固定有标签样本输入的数量，通过改变训练集中无标签样本输入的数量，研究无标签样本数量对半监督学习性能的影响实验结果如下：

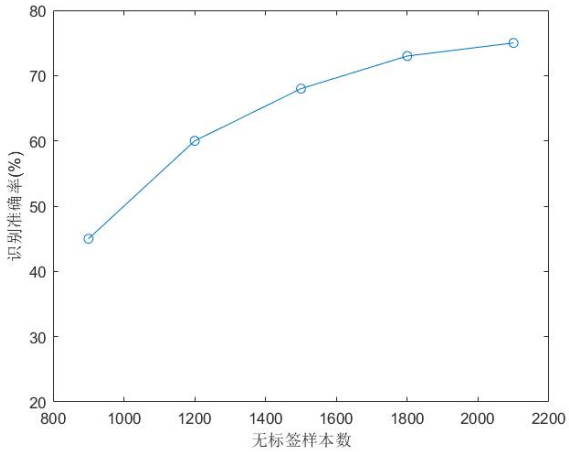


图 6-7 无标签样本数量对半监督学习性能的影响图

该实验结果表明在小样本情况下，半监督判别分析方法对脑电信号中 P300 峰的识别能力较弱，难以准确识别出目标字符。随着无标签样本数量的增加，识别准确率逐步提高，

说明增加无标签样本数能够很好的解决在有限带标签样本情况下 P300 信号的识别问题，且随着无标签样本数的增加，半监督学习的效果与第二问所得监督学习的结果相差不大。实验证明基于 S3VM 的半监督学习方法能够在保证识别精度的情况下大大减少预处理中所需标记样本的时间，从而增加 P300 脑-机系统的工作效率，为在线识别提供可能。

利用该半监督学习方法分类待识别目标(char18-char22)结果如下：

表 6-2 半监督学习方法分类待识别目标结果图

	char13	char14	char15	char16	char17	char18	char19	char20	char21	char22
s1	A	F	5	2	O	T	K	X	A	N
s2	M	D	Y	2	J	T	K	X	G	-
s3	M	L	Q	N	I	T	K	X	A	-
s4	M	4	5	2	I	T	J	X	A	0
s5	M	F	5	2	I	T	K	L	A	0

七．问题 4 的模型建立与求解

7.1 问题分析

根据题目的描述可知，人体处于不同睡眠状态时，大脑自发性产生的脑电信号存在极大差异，因而可以根据不同的脑电波形的特点，自动监测人体所处的睡眠状态。在本题中，脑电信号特征体现在它的频谱中各个区间所占的能量强度不同，我们可以根据四个分区强度分布特点来诊断人体睡眠状态。根据附件而所给出的数据特点，每个数据都带有精确无噪声的标签，监督学习分类算法能很好胜任这一类的数据。为了综合比较不同模型特点，我们提出基于随机森林，支持向量机（SVM），以及 LightGBM 三种监督学习算法来训练睡眠预测模型。

7.2 SVM 模型

7.2.1 模型建立

SVM 模型是一种基本的二分类模型，其基本思想在基于划分的训练集的样本空间中找到一个划分超平面，将不同类别的样本分开。如下图所示：

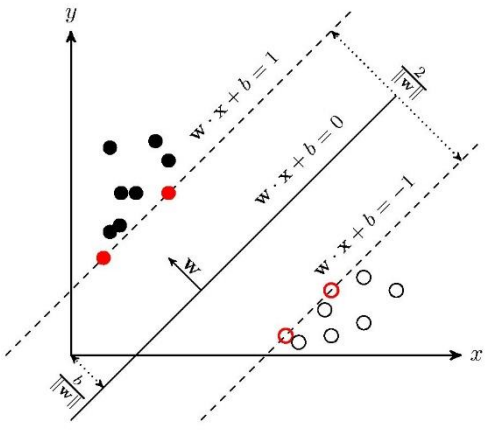


图 7-1 SVM 划分超平面图

能够划分训练数据集的超平面很多，如图中所示虚线和实现表示的三条划分超平面。

由于训练集的局限性和噪声等因素的影响，训练集外的数据可能比训练集的样本更加接近于划分超平面，这将使得许多很多超平面出现错误，SVM 的目标就是找到其中“容忍性”最好的平面以获得最鲁棒的模型。

考虑本题的随眠特征数据集，每个数据均有着四维属性的高维数据集，其训练样本空间显然不是线性可分的。对于这样一个问题，可以将样本从原始空间映射到一个更高维度的特征空间，使得样本在高维的特征空间中可分。对于有限属性的数据（如本题中的睡眠特征数据有 4 个属性），一定会存在一个特征超平面使得样本可分。

不失一般性，将一行睡眠特征数据记为x_i，则$\varphi(x_i)$即为映射后的特征向量。于是，在特征空间中划分超平面相应的模型可以表示为：

$$f(x_i) = w^T \varphi(x_i) + b \tag{36}$$

w,b为模型参数。想找到最佳超平面，模型优化等价于下列优化问题：

$$\min_{w,b} \ \frac{1}{2}||w||^2 \ \ s.t. \ y_i(w^T \varphi(x_i) + b) \geqslant 1, \ i = 1,...,m \tag{37}$$

其对偶问题为：

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \varphi(x_i)^T \varphi(x_j) \\ &s.t. \ \sum \alpha_i y_i = 0, \ \alpha_i > 0, \ i = 1,...,m \end{aligned} \tag{38}$$

求解上式涉及到计算$\varphi(x_i)^T \varphi(x_j)$，这是样本$x_i, \ x_j$特征空间向量的内积，特征向量的维度可能很高，甚至可能是无穷维。为了避开这个障碍，引入核函数

$$k(x_i,x_j) = \varphi(x_i)^T \varphi(x_j) \tag{39}$$

将其带入式（38）既可求出模型的最优解为：

$$\begin{aligned} f(x_i) &= w^T \varphi(x_i) + b \\ &= \sum \alpha_i y_i \varphi(x_i)^T \varphi(x_j) \ + \ b \\ &= \sum \alpha_i y_i k(x_i,x_j) \ + b \end{aligned} \tag{40}$$

上式也可以称为“支持向量展开式”。我们希望表征不同随眠阶段的数据在特征空间中可分，因此分类的好坏几乎由特种空间决定，而特征空间由核函数决定。在实际应用中，核函数的选择几乎决定着决定这 SVM 模型的性能，是影响模型的至关重要的因子。常见的核函数有高斯核，线性核，多项式核，RBF 等。下图给出了一个具有 3 类数据问题中，他们训练出的分类超平面的示意图。

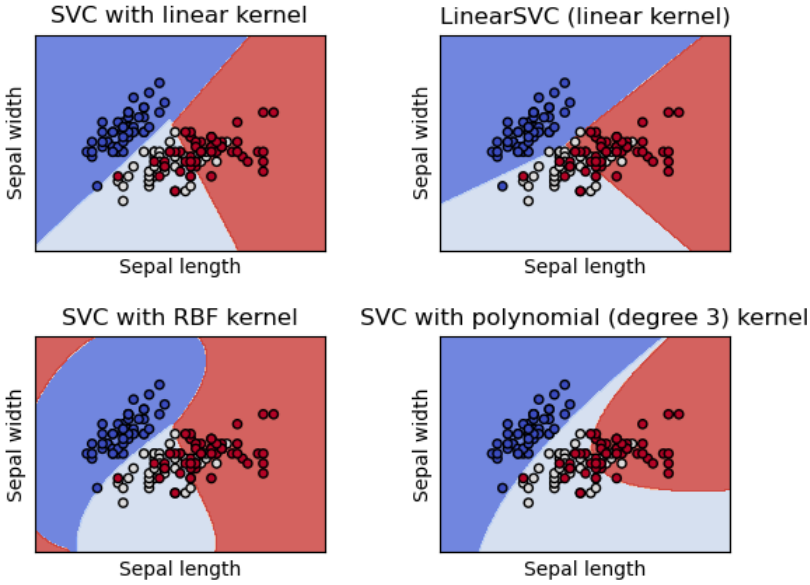


图 7-2　SVM 模型训练下分类超平面示意图

一般情况下，4 种核函数分类结果是可互相比拟的，但考虑到模型训练的速度和稳定行，在本题目中，我们选择常见线性内核，即

$$k(x_i, x_j) = {x_i}^T x_j \tag{41}$$

以求得最佳模型。

在预测测试集数据时，将随眠特征数据带入式（36），按照函数值的大小判断其所属的类别。

另外一个需要解决的问题是，SVM 通常只支持两类标签数据的分类，即 0，1 分类。通常实现对多标签数据分类支持有三种方法，分别为“one vs one”，“one vs others”和“many vs many”。本题中，我们采用”one virus one”方法。具体做法是每次选择 2 个表征随眠状态数据训练出一个分类器，这样需要训练出 $\frac{m(m-1)}{2}$ 个分类器。在预测结果时，最终的睡眠状态分类结果采用投票机制得出，即得分最多的类别即为数据所属于的类别。

7.2.2 模型求解

我们使用 python 语言搭建并训练所提出的模型。首先，我们将题目所给 5 组表征不同的睡眠状态的数据分别分离出它们的标签和睡眠特征数据集合，然后按照顺序，将表征状态的数据和标签数据按照顺序组成一个综合状态数据集和对应标签数据集合。为了达到更好的模型训练效果，采用随机函数，重新生成位置排列，并且按照生成位置排列，打乱对应的状态数据集合和标签数据集合，为方便说明，我们将整体睡眠数据集合记为 D。

其次一个重要的问题是测试集和训练集合的选取问题。最直接的做法即为直接从 D 中选取一定比例的数据作为测试集，其余为训练集（这种方法也成为留出法）。但留出法的缺陷就是一定比例的训练集该怎样从数据集中分割出来。为了克服这一弊端，我们采用“交叉验证法”划分训练集和数据集合，具体做法为将数据集合分成 k 个互斥的子集，每个集合各类睡眠状态比例分布与总体保持一致，每次选取 $k-1$ 个数据集作为训练集，剩下作为测试集，这样获得 $k-1$ 个训练集/测试集。本题目中，我们选取不同的训练集和测试集比例，每个比例训练 5 次，取他们平均精度作为本次的训练模型的精度。下图为模型测试集精度随着测试集样本占全体样本比例的变化情况。

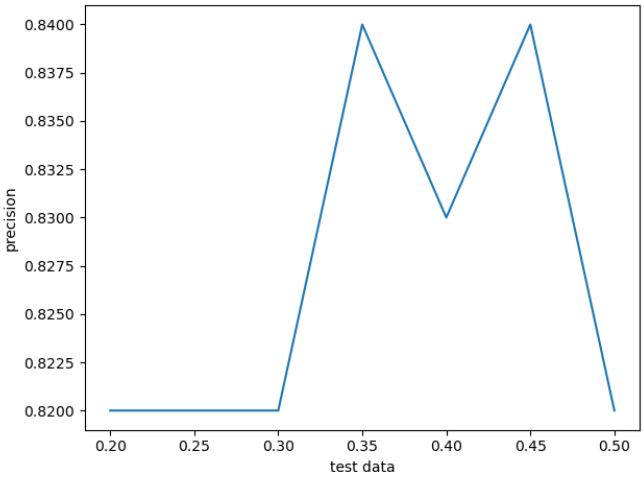


图 7-3 SVM 模型测试集精度占全体样本比例图

从图中可以看出，图线整体呈现先上升后下降的趋势，在训练样本占总体样本 70%时候的训练精度甚至不如训练样本占 65%时候的模型精度，这是由于模型的过拟合导致。在训练样本下降至总体样本的 50%及其以下时，泛化误差迅速增加，模型出现欠拟合，这是由于数据缺乏多样性能导致模型训练精度下降。依据我们的实验，我们选择35%的数据作为测试集，65%数据作为训练集，此时模型预测的睡眠精度到达最高，达到84%。此时各种睡眠状态聚类的示意图如下：

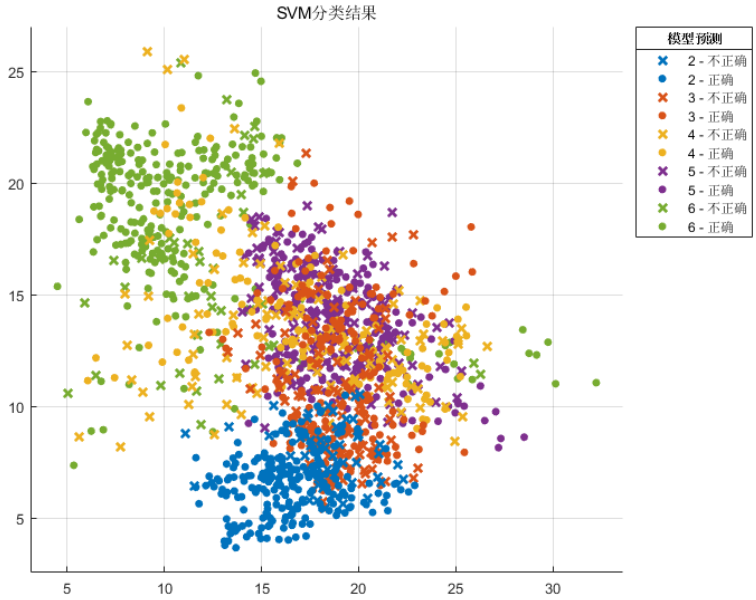


图 7-4 各种睡眠状态聚类图

其中不同颜色代表不同睡眠状态的分类结果，对应情况如下表所示，圆圈代表分类成功的数据，叉代表分类失败的数据。

表 7-1 不同睡眠状态表

红色	绿色	黄色	紫色	蓝色
清醒期	睡眠 I 期	睡眠 II 期	深睡眠期	快速眼动期

7.3 LGBM 模型

7.3.1 模型建立

LightGBM 是是一个使用基于树的学习算法的梯度增强框架。传统的梯度决策树在训练睡眠状态预测模型时存在一个难以解决的问题。即如何减少训练数据问题，常用的方法一般都是通过降低采样的方式进行，如 SGB。SGB 在每轮迭代中用随机子集训练弱学习器，并且采样率在训练过程中需要动态调整。所有这些工作还需要借助于 AdaBoost，并且由于一般的 GBDT 没有数据实列的权重，所以不能直接运用到 GBDT 上。SGB 是可以运用到 GBDT 上的，但会带来模型预测精确度下降这一致命缺点。

直方图优化：传统的基于树的决策学习算法如 XGBoost 中采用，计算过程中按照值排序，逐个数据样本来划分收益，这种算法简单但无法优化且没有很好的推广性。LightGBM 采用直方图算法，将每一个值划分到一系列的离散域中，数据的表达更加简化，能够减少内存使用，直方图也带来一定的正则化效果，能够避免模型产生过拟合。直方图优化的细节处理如下图 7-5 图 7-6 所示：

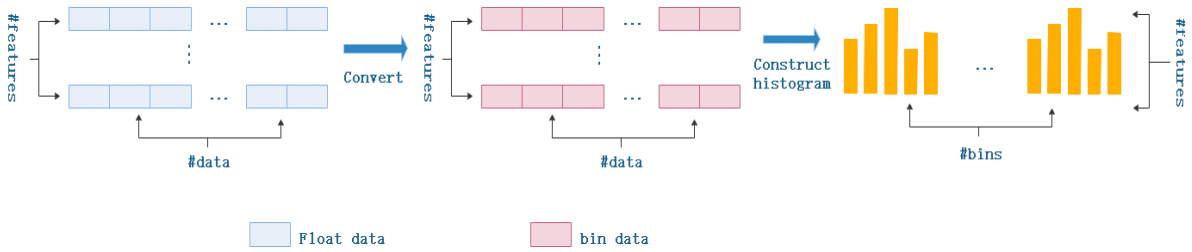


图 7-5 直方图优化示意图

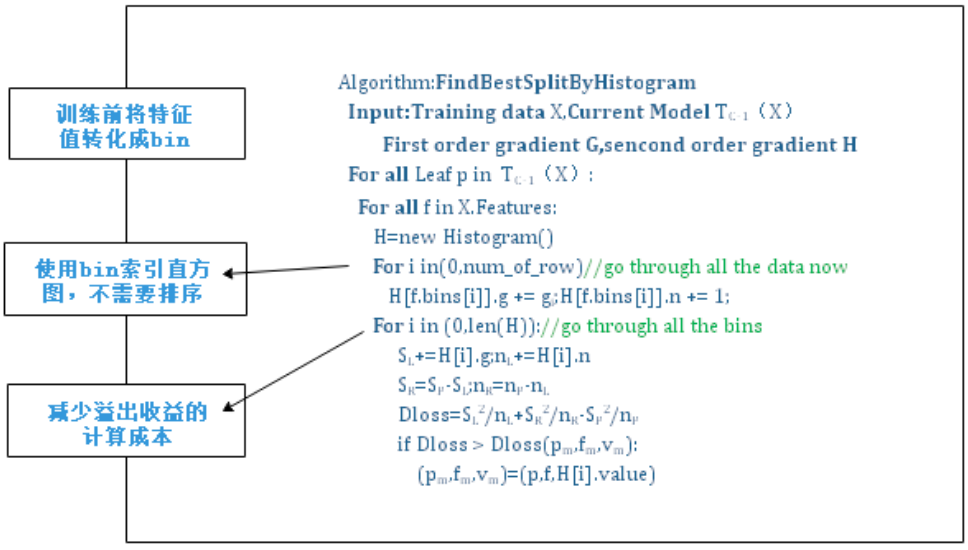


图 7-6 直方图优化算法

准确率的优化：大多数决策时采用“Level-Wise”树生长策略，如图所示。算法不加区分的对待每一层叶子，并且由于很多叶子的分裂收益较低，没有必要进行搜索和分裂，因此本方法存在大量没有必要的开销。为了克服这一缺点，LGBM 采用另外一种称为

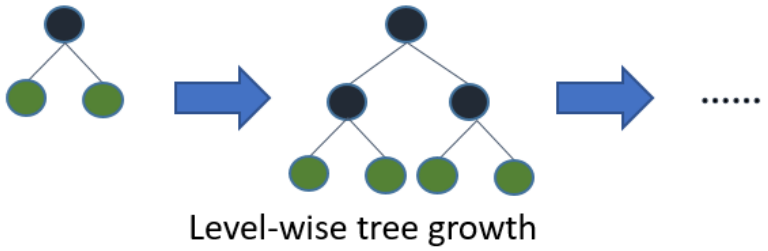


图 7-7 “Level-Wise”树生长策略图

“Leaf-Wise”的更为高效的生长策略。每次都从当前所有叶子中找到分裂增益最大的叶子，然后分裂，如此循环。与“Level-Wise”相比，可以降低更多的误差。但同样，“Leaf-Wise”可能会产生比较深的决策树，产生过拟合问题。此问题可以通过增加一个最大树最大深度限制来缓解。

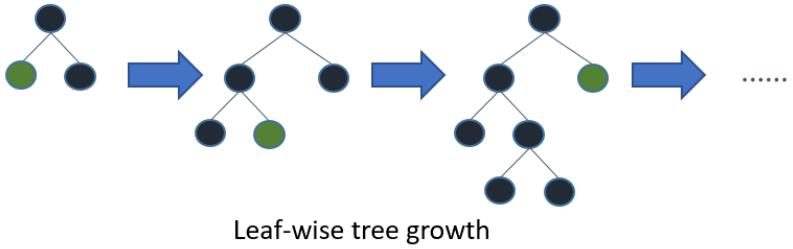


图 7-8 最大树最大深度示意图

7.3.2 模型求解

在求解睡眠状态预测模型时，首先要做的第一件事情就是模型超参数的设置。因为 LGBM 是基于“Leaf-Wise”算法的，需要设置超参数时模型的叶子树而不是基于一般决策树算法所设置的树的深度。一般情况下二者满足 $num\_leaves = 2 \times num\_depth$ 这一简单关系，本题中，我们发现设置为 300 是最为合适的状态。同样了为了减轻模型的过拟合问题，我们设置树的最大深度为 10。梯度下降的学习速率设置为 0.01。最后需要设置预测模型能够进行多标签分类。

同 SVM 模型一样，我们同样使用“交叉验证的方法”来分割训练集和测试集。每次选取不同数量比例的数据作为测试集，每轮实验进行 5 次，取预测精度平均值作为本轮实验的结果。需要注意的是，LGBM 模型给出的是每个分类标签的后验置信概率，我们取概率最大的标签作为本睡眠特征数据的睡眠状态。下图为模型测试集精度随着测试集样本占全体样本比例的变化情况。

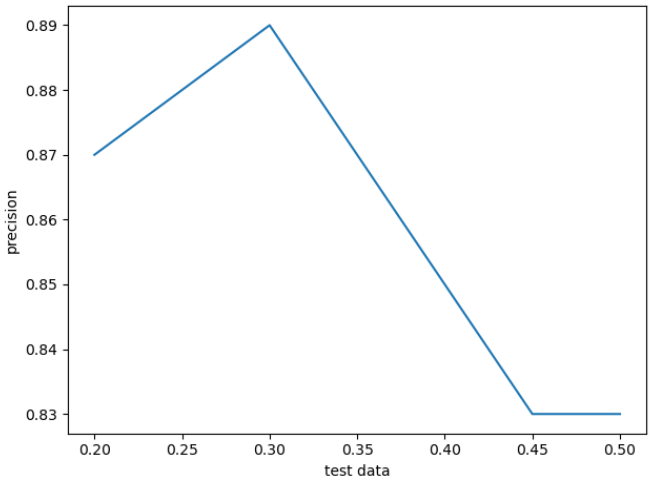


图 7-9 LGBM 模型测试集精度占全体样本比例图

上图曲线走势和使用 SVM 模型训练时几乎相同，LGBM 模型在选取 30%睡眠特征数据作为样本时，测试精度达到最高值 89%，而选取低于或高于 30%比例时，模型的预测精度均出现下降，选取较多的训练数据，会导致模型过拟合，泛化性能迅速降低；较少的训练数据，模型的泛化误差也会增大，本数据集合最合适的训练集样本数量在 75%左右，此时为最优模型，分类结果图示如下, 不同颜色与不用睡眠状态对应关系见表（7-1）。

图 7-10　LGBM 模型下分类结果图

7. 4 随机森林模型

最后一个模型，我们选取问题一中所展示的随机森林模型，并与前面的两种模型结果进行对比。同样采取“交叉验证”的方法分割睡眠特征数据集，每次选取不同比例的训练样本，每次实验重复 5 轮，取平均精确度作为实验结果。下图分别表示测试集精度随着测试集样本占全体样本比例的变化情况和选取最佳比例的测试数据数量时候的分类结果：

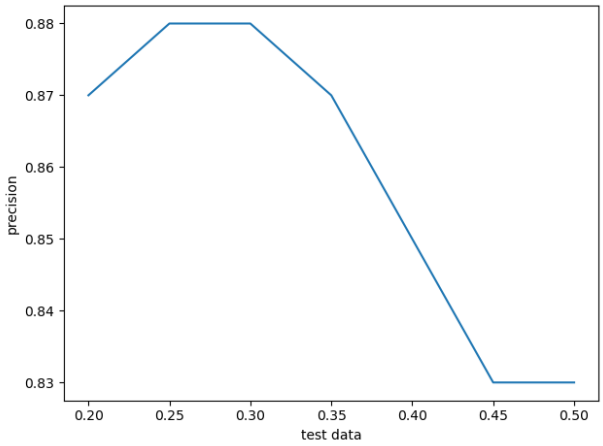


图 7-11 LGBM 模型测试集精度占全体样本比例图

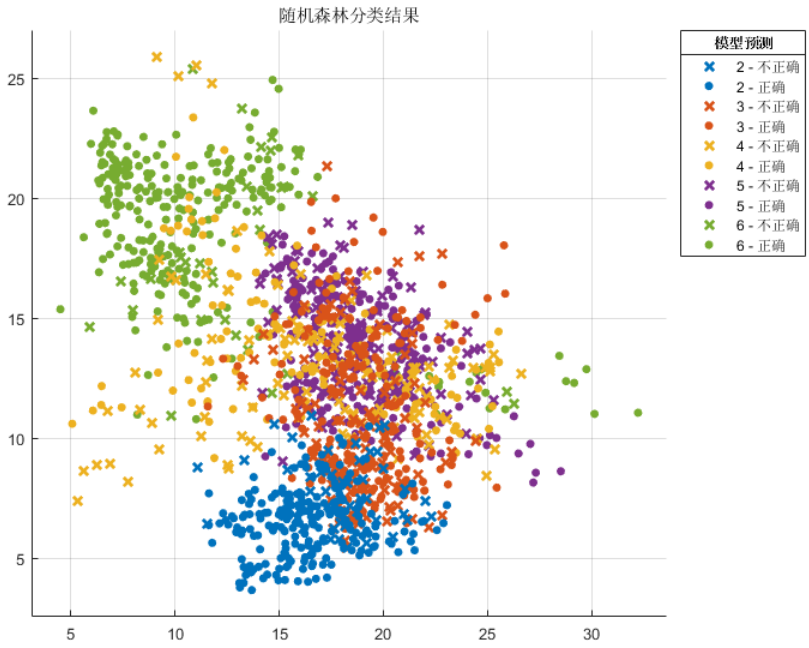


图 7-12 随机森林模型下分类结果图

三种方法综合起来比较，从预测精度来看，随机森林和 LGBM 算法性能互相接近，并且都要明显由于基于 SVM 的分类方法。均选取 30%测试数据表现看，随机森林和 LGBM 都达到了 89%的预测精度，而 SVM 仅能达到 84%。并且达到相同的睡眠状态精度，随机森林和 LGBM 算法需要更少的观测数据。为了比较三种算法的时间复杂度，我们均在选取了 30%比例测试数据情况下，测试算法运行的时间，为了消除随机扰动影响，每个算法跑 10 轮取平均时间作为最终的实验结果。结果表明 SVM 算法最快，其次 LGBM，最慢为随机森林，结合模型的预测准确度来看，LGBM 算法的综合性能是最优的。三种算法的运行时间和精度如下表所示：

表 7-2 三种算法比较表

算 法	SVM	LGBM	随机森林
时 间（s）	0.0193	0.0210	0.0229
精 度（%）	84	89	89

8．模型的总结与评价

8.1 模型的优点

在问题一中，我们通过数据简化，标签化处理，生成训练集，建立了基于随机森林算法的分类模型。利用 python 在使用测试数据 5 轮的情况下成功得到被试的 10 个测试结果（共 48 个目标），与问题而所给的 5 个测试目标验证后，得到的目标字符数据具有很好精度和合理性。

在问题二中，我们首先对数据进行滤波预处理，充分考虑到脑电信号的子集的空间稀疏性，因此建立“spike and slab”稀疏贝叶斯回归模型，进一步分析出对所有被试者都适用的最有通道组合。依据通道权重选择出的最优通道组合，不仅大大减少了数据冗余，噪声数据，还保持了模型的预测精度。

在问题三中，我们对最优通道选择的数据进行滤波预处理，将训练集分为标签数据集和无标签数据集。选取半监督学习的 S3VM 分类模型，并引入正负样本比例因子 r 排除正负样本数目不平衡的问题。在训练时对初始无标签样本和有标签样本赋予不同的权重，最终在验证集上得到了较高的正确率，与在测试数据中得到的结果基本吻合。该模型在保证预测精度的前提下节省了人工标注训练集的时间，提高了 P300 系统的运行效率。

在问题四中，我们综合比较了基于随机森林，支持向量机（SVM）, 以及 LightGBM 三种监督学习算法来训练睡眠预测模型。同时在测试集和训练集合的选取问题上，我们采用了“交叉验证法”，它克服因为训练集测试集不同分割方法引起模型性能不稳定。测综合比较 SVM，随机森林和 LGBM 算法预测精度和时间复杂度，得出 LGBM 算法得到的精度较高，且算法测试运行时间较短，其综合运行性能最优。

8.2 模型的缺点

由于数据文件中数据量巨大，所以在数据预处理的过程中可能会存在遗漏丢失的现象，同时在数据筛选过程中可能会有方法不太合理，这也对结果造成了一定的影响。在后续深入研究学习中，还应当在保证分类准确性的情况下，进一步提升信息的传输速率，并应用到实际的 P300-BCI 系统中去。

9. 参考文献

[1] Zhao B , Wang F , Zhang C . CutS3VM: A fast semi-supervised SVM algorithm[C]// Acm Sigkdd International Conference on Knowledge Discovery & Data Mining. ACM, 2008.

[2] Li Y F , Zhou Z H . Improving Semi-Supervised Support Vector Machines Through Unlabeled Instances Selection[J]. 2010.

[3] Ogawa K, Imamura M, Takeuchi I, et al. Infinitesimal annealing for training semi-supervised support vector machines[C]//International Conference on Machine Learning. 2013: 897-905.

[4] Andersen M R, Winther O, Hansen L K. Bayesian inference for structured spike and slab priors[C]//Advances in Neural Information Processing Systems. 2014: 1745-1753.

[5] Scheipl F, Fahrmeir L, Kneib T. Spike-and-slab priors for function selection in structured additive regression models[J]. Journal of the American Statistical Association, 2012, 107(500): 1518-1532.

[6]杨雅茹. 基于脑电信号的癫痫发作预测算法研究[D].天津职业技术师范大学,2020.

[7]康乾坤,路来君.随机森林算法在测井岩性分类中的应用[J].世界低质，2020，39（02）： 398-405.

[8] Park T, Casella G. The bayesian lasso[J]. Journal of the American Statistical Association, 2008, 103(482): 681-686.

[9] Meier L, Van De Geer S, Bühlmann P. The group lasso for logistic regression[J]. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 2008, 70(1): 53-71.

[10] 任威风. 面向便携式脑机接口的 P300 模式增强与识别研究[D].厦门大学,2017.

[11] 李宏伟. 基于 P300 的脑机接口拼写器的优化与实践[D].浙江大学,2018.

[12] Selim A E, Wahed M A, Kadah Y M. Machine learning methodologies in P300 speller Brain-Computer Interface systems[C]//2009 National Radio Science Conference. IEEE, 2009: 1-9.

[13] Kaper M, Meinicke P, Grossekathoefer U, et al. BCI competition 2003-data set IIb: support vector machines for the P300 speller paradigm[J]. IEEE Transactions on biomedical Engineering, 2004, 51(6): 1073-1076.

[14] Kshirsagar G B, Londhe N D. Improving performance of Devanagari script input-based P300 speller using deep learning[J]. IEEE Transactions on Biomedical Engineering, 2018, 66(11): 2992-3005.

[15] Hoffmann U, Yazdani A, Vesin J M, et al. Bayesian feature selection applied in a P300 brain-computer interface[C]//2008 16th European Signal Processing Conference. IEEE, 2008: 1-5.

[16] 王俊杰. P300 脑机接口的半监督和无监督学习算法研究[D].华南理工大学,2017.

[17] 张锦涛. P300 脑机接口的在线半监督学习算法与系统研究[D].华南理工大学,2015.

10. 附录

1. 滤波器设计代码（matlab）：

```
function Hd = bandpass
%BANDPASS Returns a discrete-time filter object.
% Butterworth Bandpass filter designed using FDESIGN.BANDPASS.

% All frequency values are in Hz.
Fs = 250;  % Sampling Frequency

Fstop1 = 0.1;         % First Stopband Frequency
Fpass1 = 0.2;         % First Passband Frequency
Fpass2 = 20;          % Second Passband Frequency
Fstop2 = 30;          % Second Stopband Frequency
Astop1 = 80;          % First Stopband Attenuation (dB)
Apass  = 1;           % Passband Ripple (dB)
Astop2 = 80;          % Second Stopband Attenuation (dB)
match  = 'stopband';  % Band to match exactly

% Construct an FDESIGN object and call its BUTTER method.
h  = fdesign.bandpass(Fstop1, Fpass1, Fpass2, Fstop2, Astop1, Apass, ...
                      Astop2, Fs);
Hd = design(h, 'butter', 'MatchExactly', match);

% [EOF]
```

2. 随机森林预测模型（python 3.8）：

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import numpy as np
import scipy.io as sio

#一号被试预测模型
#数据导入
file = sio.loadmat('train.mat')
s1_tem = file['S1_train']

file1 = sio.loadmat('trainevent.mat')
s1_eve = file1['S1_train_event']

file2 = sio.loadmat('testEVENT.mat')
s1_test_tem = file2['S1_test']
```

```
18.
19. file3 = sio.loadmat('test.mat')
20. s1_test_eve = file3['S1_test_event']
21.
22. # 训练 1 号被试者
23. dict={'0':[1,8],'2':[1,10],'4':[2,7],'6':[2,12],'8':[3,9],'10':[3,11],'12':[4,7],'14':[
    4,10],'16':[5,8
24. ],'18':[5,12],'20':[6,9],'22':[6,11]}
25. labels = []
26. train_data = np.zeros((60*12,125))
27. index = 0
28. for i in range(0,24,2): #列
29.     for j in range(66): #行
30.         h = s1_eve[j][i]
31.         if h<100:
32.             be = s1_eve[j][i+1]
33.             train_data[index,:] = s1_tem[be-1:be+124,i//2]
34.             if h in dict[str(i)]:
35.                 labels.append(1)
36.             else:
37.                 labels.append(0)
38.             index+=1
39.
40. labels = np.array(labels)
41. scaler = StandardScaler() # 标准化转换
42. scaler.fit(train_data)  # 训练标准化对象
43. train_data= scaler.transform(train_data)
44.
45. x_train, x_test, y_train, y_test = train_test_split(train_data, labels)
46. clf = RandomForestClassifier()
47. clf.fit(x_train, y_train)
48. pres = clf.predict(x_test)
49. print(accuracy_score(pres,y_test))
50. # 预测一号被试者
51. predict_data = np.zeros((60*10,125))
52. index = 0
53. for i in range(0,20,2): #列
54.     for j in range(66): #行
55.         h = s1_test_eve[j][i]
56.         if h<100:
57.             be = s1_test_eve[j][i+1]
58.             predict_data[index,:] = s1_test_tem[be-1:be+124,i//2]
59.             index+=1
60.
```

```
scaler = StandardScaler() # 标准化转换
scaler.fit(predict_data)  # 训练标准化对象
predict_data= scaler.transform(predict_data)

res1 = clf.predict(predict_data)
vote = np.zeros((10, 13))
index = 0
for i in range(0,20,2): #列
    for j in range(66): #行
        h = s1_test_eve[j][i]
        if  h<100:
            if res1[index]:
                vote[i//2,h] +=1
            index+=1
```

3. 基于 EP 的稀疏贝叶斯回归代码实现（python3.8）

```
import numpy as np

import matplotlib.pyplot as plt
import scipy.io as sio
def reconstruct(A, y, P, K, sigma02, nvi2, tau0, alpha):
    ATA, ATy = np.dot(A.T, A), np.dot(A.T, y)
    J, H = ATA / sigma02, ATy / sigma02
    # Initialize Global Approximation
    mi, sigmai, nvi = np.zeros(A.shape[1]), np.ones(A.shape[1]), 0.5 *
np.ones(A.shape[1])
    mi_old = mi
    nvi_old = nvi

    # Initialize Site Approximation
    mi2_sigmai2_inv, sigmai2_inv, nvi1 = np.zeros(A.shape[1]), 1e-6 *
np.ones(A.shape[1]), 0.5 * np.ones(K)
    for ittcs in range(max_itt):
        # compute cavity
        mi_sigmai_inv_bar, sigmai_inv_bar = mi / sigmai - mi2_sigmai2_inv, 1 /
sigmai - sigmai2_inv
        nvi_bar = nvi2

        # positive cavity variance
        index = sigmai_inv_bar < 0
        update_index = sigmai_inv_bar > 0
        sigmai_inv_bar[index] = 1

        # Compute matching moments
```

```
        mi_new, vi_new, nvi1_new
 =
compute_cs_spike_and_slab_moments(mi_sigmai_inv_bar, sigmai_inv_bar,
nvi_bar, tau0)

        # Compute Site Updates
        sigmai2_new_inv = 1 / vi_new - sigmai_inv_bar
        mi2_sigmai2_inv_new = mi_new / vi_new - mi_sigmai_inv_bar
        idx = sigmai2_new_inv <= 0
        sigmai2_new_inv[idx] = 1e-2
        sigmai2_new_inv = bound_values(sigmai2_new_inv, MINIMUM_VARIANCE
, MAXIMUM_VARIANCE)

        # adjust vi_new and sigmai2_new_inv
        vi_new[idx] = 1 / (sigmai2_new_inv[idx] + sigmai_inv_bar[idx])
        mi2_sigmai2_inv_new[idx] = mi_new[idx] / vi_new[idx]
 - mi_sigmai_inv_bar[idx]

        # update with damping
        nvi1[update_index] = (1 - alpha) * nvi1[update_index] + alpha *
 nvi1_new[update_index]
        sigmai2_inv[update_index] = (1 - alpha) * sigmai2_inv[update_index] +
 alpha * sigmai2_new_inv[update_index]
        mi2_sigmai2_inv[update_index] = (1 - alpha) *
mi2_sigmai2_inv[update_index] + alpha * mi2_sigmai2_inv_new[
            update_index]

        # Update global approximation
        mi, sigmai, nvi = update_global(A, P, K, sigma02, mi2_sigmai2_inv,
sigmai2_inv, J, H, nvi1, nvi2)
        # Check for EP convergence
        ep_diff = np.linalg.norm(mi - mi_old) / np.linalg.norm(mi)
        nvi_dif = np.linalg.norm(nvi - nvi_old) / np.linalg.norm(nvi)
        if ep_diff < 1e-5 and nvi_dif < 1e-5:
            break
        mi_old = mi
        nvi_old = nvi
return mi
def update_global(A, P, K, sigma02, mi2_sigmai2_inv, sigmai2_inv, J, h, nvi1, nvi2,
diagnoal_only=True):
    if P > K:  # compute directly
        p1 = J + np.diag(sigmai2_inv)
        mi, sigmai = np.linalg.solve(p1, h + mi2_sigmai2_inv), np.linalg.inv(p1)

```

```
    else:  # Use Woodburys identity
        w = A / sigmai2_inv # equal to A*diag(sigmai2)
        r = np.linalg.solve(sigma02 * np.identity(P) + np.dot(A, w.T), w)
        wtr = np.dot(w.T, r)

        # Only compute diagonal of posterior covariance
        if diagnoal_only:
            mi = (h + mi2_sigmai2_inv) / sigmai2_inv - np.dot(wtr, h +
mi2_sigmai2_inv)
            sigmai = 1 / sigmai2_inv - np.sum(w * r, 0)

        else:
            mi, sigmai = (h + mi2_sigmai2_inv) / sigmai2_inv - np.dot(wtr, h +
mi2_sigmai2_inv), np.diag(1 / sigmai2_inv) - wtr
    nvi = 1 /(1 + (1-nvi1) * (1 - nvi2) / nvi1 / nvi2)
return mi, sigmai, nvi
def compute_cs_spike_and_slab_moments(mi_sigmai_inv_bar,
sigmai_inv_bar, nvi_bar, tau0):
    r = log_npdf(0, mi_sigmai_inv_bar /sigmai_inv_bar,
1/sigmai_inv_bar) - log_npdf(0,
 mi_sigmai_inv_bar /
sigmai_inv_bar, 1 /
sigmai_inv_bar + tau0)
    nvi1_new = 1 / (1 + np.exp(r))
    w = np.log(1 -nvi_bar) - np.log(nvi_bar) + r
    c = bound_values(1 + np.exp(w), MINIMUM_VARIANCE,
MAXIMUM_VARIANCE)
    denom = sigmai_inv_bar + 1 / tau0
    mi_new = mi_sigmai_inv_bar / denom / c
    aux = (1 +(mi_sigmai_inv_bar)**2/denom)/denom / c
    vi_new = aux - mi_new**2
    vi_new = bound_values(vi_new, MINIMUM_VARIANCE, MAXIMUM_VARIANCE)
    return mi_new, vi_new, nvi1_new

P = 40
K = 128
sigma02 = 0.01
tau0 = 1
alpha = 0.8

file = sio.loadmat('train.mat')
D = file['dict']
Y = file[‘label]

```

```
117.mi = fc.reconstruct(D, y, P, K, sigma02, nvi2,
118.tau0, alpha)
```

4. LGBM SVM 算法（python）

```
1.  import numpy as np
2.  from sklearn import svm
3.  from sklearn.ensemble import RandomForestClassifier
4.  from sklearn.metrics import precision_recall_curve
5.  from sklearn.metrics import accuracy_score
6.  from sklearn.metrics import classification_report
7.  from sklearn.model_selection import train_test_split
8.  import scipy.io as sio
9.
10. # load data
11. file = sio.loadmat('sleep.mat')
12. print(file.keys())
13. m6_data = file['m6a']
14. m5_data = file['m5a']
15. m4_data = file['m4a']
16. m3_data = file['m3a']
17. m2_data = file['m2a']
18.
19. # split labels and train_set
20. label_2 = m2_data[:,0]
21. label_3 = m3_data[:,0]
22. label_4 = m4_data[:,0]
23. label_5 = m5_data[:,0]
24. label_6 = m6_data[:,0]
25.
26. train_data_2 = m2_data[:,1:]
27. train_data_3 = m3_data[:,1:]
28. train_data_4 = m4_data[:,1:]
29. train_data_5 = m5_data[:,1:]
30. train_data_6 = m6_data[:,1:]
31.
32. #split train and test data
33. x_train_2, x_test_2, y_train_2, y_test_2 = train_test_split(train_data_2, label_2,test_
    size=0.3)
34. x_train_3, x_test_3, y_train_3, y_test_3 = train_test_split(train_data_3, label_3,test_
    size=0.3)
35. x_train_4, x_test_4, y_train_4, y_test_4 = train_test_split(train_data_4, label_4,test_
    size=0.3)
36. x_train_5, x_test_5, y_train_5, y_test_5 = train_test_split(train_data_5, label_5,test_
    size=0.3)
```

```
x_train_6, x_test_6, y_train_6, y_test_6 = train_test_split(train_data_6, label_6,test_size=0.3)

# conbine trian and test data
train_data = np.vstack((x_train_2,x_train_3,x_train_4,x_train_5,x_train_6))
trian_label = np.hstack((y_train_2,y_train_3,y_train_4,y_train_5,y_train_6))
test_data = np.vstack((x_test_2,x_test_3,x_test_4,x_test_5,x_test_6))
test_label = np.hstack((y_test_2,y_test_3,y_test_4,y_test_5,y_test_6))

state1 = np.random.get_state()
np.random.shuffle(train_data)
np.random.set_state(state1)
np.random.shuffle(trian_label)

state2 = np.random.get_state()
np.random.shuffle(test_data)
np.random.set_state(state2)
np.random.shuffle(test_label)

clf = svm.SVC(decision_function_shape='ovo')
clf.fit(train_data,trian_label)
predict_res = clf.predict(test_data)

print('SVM accuracy')
print(accuracy_score(predict_res,test_label))

clf2 = RandomForestClassifier()
clf2.fit(train_data,trian_label)
res2 = clf2.predict(test_data)
print('Rondom_Forest')
print(accuracy_score(res2,test_label))

# create dataset for lightgbm
lgb_train = lgb.Dataset(X_train, y_train)
lgb_eval = lgb.Dataset(X_test, y_test, reference=lgb_train)
# specify your configurations as a dict
params = {
    'boosting_type': 'gbdt',
    'objective': 'multiclass',
    'num_class': 5
    'metric': 'multi_error',
    'num_leaves': 300,
    'min_data_in_leaf': 100,
    'learning_rate': 0.01,
```

```
    'feature_fraction': 0.8,
    'bagging_fraction': 0.8,
    'bagging_freq': 5,
    'lambda_l1': 0.4,
    'lambda_l2': 0.5,
    'min_gain_to_split': 0.2,
    'verbose': 5,
    'is_unbalance': True
}
# train
print('Start training...')
gbm = lgb.train(params,
                lgb_train,
                num_boost_round=10000,
                valid_sets=lgb_eval,
                early_stopping_rounds=500)

print('Start predicting...')

preds = gbm.predict(test_x, num_iteration=gbm.best_iteration)  # 输出的是概率结果'''
```